UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

TRANSMISIÓN DE SECUENCIAS DE VÍDEO A TASA BINARIA MUY BAJA Y ADAPTABLE BASADA EN GENERACIÓN Y TRANSMISIÓN DE DESCRIPCIONES

-PROYECTO FIN DE CARRERA-

Juan Carlos San Miguel Avedillo
Septiembre de 2006

TRANSMISIÓN DE SECUENCIAS DE VÍDEO A TASA BINARIA MUY BAJA Y ADAPTABLE BASADA EN GENERACIÓN Y TRANSMISIÓN DE DESCRIPCIONES

AUTOR: Juan Carlos San Miguel Avedillo
TUTOR: José María Martínez Sánchez

Grupo de Tratamiento de Imágenes
Dpto. de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Septiembre de 2006

PROYECTO FIN DE CARRERA

Título: *Transmisión De Secuencias De Vídeo A Tasa Binaria Muy Baja y Adaptable Basada En Generación y Transmisión De Descripciones*

Autor: D. Juan Carlos San Miguel Avedillo

Tutor: D. José María Martínez Sánchez

Tribunal:

Presidente: Jesús Bescós Cano

Vocal: Pedro Pascual Broncano

Vocal secretario: José María Martínez Sánchez

Fecha de lectura:

Calificación:

Palabras clave

Segmentación, Adaptación, Vídeo, Descripciones, Vídeo-seguridad, MPEG-7

Resumen

El principal objetivo de este PFC es el diseño de un sistema para transmitir información relevante de secuencias de cámaras de seguridad a muy baja tasa binaria y el desarrollo correspondiente para demostrar su viabilidad. Las secuencias se analizan para detectar las zonas de actividad (análisis de movimiento) y diferenciarlas del fondo (que se transmite solamente cuando éste tenga cambios significativos, periódicamente, o bajo demanda) y se transmite, en lugar o adicionalmente a los objetos de vídeo, descripciones de las regiones en movimiento. El sistema también proporciona adaptación de las secuencias captadas a las capacidades de diversos terminales y redes de comunicación.

Abstract

The main objective of this PFC is the design of a system to transmit relevant information of security sequences (from a static camera) to very low binary rate and the corresponding development to demonstrate its viability. The sequences are analyzed to detect the zones of activity (movement analysis) and to differentiate them from the bottom (that is only transmitted when this one has significant changes, periodically, or on-demand) and is transmitted, in place or additionally to the objects of video, descriptions of the regions in movement. The system also provides adaptation of the sequences caught to the capacities of diverse terminals and communications networks.

Ese proyecto ha sido parcialmente financiado por la cátedra Infoglobal-Uam para "Nuevas tecnologías de video aplicadas a sistemas de seguridad"

Agradecimientos

En primer lugar quería agradecer a mi tutor, José María Martínez Sánchez, por asignarme este proyecto tan interesante, y por estar siempre dispuesto a atenderme con buen humor.

Además, quería agradecer la ayuda que me han prestado los miembros del GTI. Especialmente quería destacar la ayuda de Luis, Herranz, Víctor Valdés, Javier Molina, Víctor Fernández y Álvaro García. Cada uno de alguna manera me ha ayudado a la conclusión de este proyecto.

A mi compañero Miguel, que me ha ofrecido siempre su valiosa ayuda durante toda la carrera.

También quería agradecer a Jesús Bescós la continua ayuda que nos ha prestado a lo largo de toda la carrera a todos mis compañeros de promoción y a mí.

Y en especial a mis padres, por su apoyo y confianza durante esta etapa en la universidad.

Juan Carlos San Miguel Avedillo
Septiembre de 2006.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

1 Introducción

1.1 Motivación

La motivación de este PFC es aplicar a un sistema de vídeo-seguridad[1] las tecnologías de sistemas de generación y transmisión de descripciones para lograr menores tasas binarias (a cambio de mayor necesidad de proceso en el emisor y receptor: para análisis y síntesis, respectivamente) y adaptación de las secuencias captadas a diversos terminales y condiciones de uso (e.g., tasa binaria). Estas tecnologías serán parte de la base de las futuras aplicaciones de video-seguridad, en las cuales el conocimiento de lo que ocurre en las secuencias captadas permitirá tener un mayor control y discriminación sobre la ingente cantidad de información existente en cualquier sistema de video-seguridad.

Este Proyecto Fin de Carrera (PFC) utiliza principalmente el estándar *MPEG-7*[2] que proporciona una forma estandarizada para describir contenidos multimedia.

MPEG-7 define una serie de herramientas de descripción para crear descripciones (metadatos) del contenido audiovisual, tratando de dar solución, entre otros, al problema de la gestión de la ingente cantidad de contenido audiovisual disponible en la red.

1.2 Objetivos

El principal objetivo de este PFC es el desarrollo de un sistema para transmitir información relevante de secuencias de cámaras de seguridad a muy baja tasa binaria. Las secuencias se analizan para detectar las zonas de actividad (análisis de movimiento) y diferenciarlas del fondo (que se transmitirá solamente cuando éste tenga cambios significativos, periódicamente, o bajo demanda) y se transmite, en lugar o adicionalmente a los objetos de vídeo, descripciones de las regiones en movimiento. Estas descripciones se pueden generar a diversos niveles de calidad (detalle) variable en función de las prestaciones del sistema, pudiendo ir desde la trayectoria del centroide del objeto en movimiento hasta la trayectoria de la silueta detallada del objeto, pasando por regiones rectangulares que incluyan al objeto (“bounding box”). La región puede llevar asociada una textura (imagen) para mejorar las prestaciones. La no presentación de dicha textura permite proporcionar aspectos de privacidad en la mayoría de los casos, revelándose la personalidad del objeto en movimiento solamente en los casos necesarios.

Adicionalmente a los algoritmos de análisis de imágenes en el terminal emisor (detección de objetos, generación de fondo,…) y la generación de las descripciones y los objetos de vídeo asociado (vídeo anotado[3]), se han desarrollado diversas técnicas de visualización de las descripciones (y objetos de vídeo asociados) en función de las capacidades del medio de transmisión y del terminal receptor.

En resumen, los objetivos principales del PFC son:

- Desarrollo de un aplicación transmisora que realiza las siguientes tareas
    - Generación de algoritmos capaces de separar fondo y objetos (en movimiento y estáticos) de una secuencia de imágenes
    - Generación de descripciones del contenido que aparece en la secuencia de imágenes en MPEG-7 de la información que se desea transmitir.
    - Creación de diversos niveles de descripciones y contenidos visual asociado, para adaptarlo a diversas capacidades del sistema (terminal y red) y requisitos de la aplicación final.
- Desarrollo de una aplicación receptora que realiza las siguientes tareas
    - Síntesis y visualización de las diversas descripciones recibidas

La etapa de transmisión/recepción de las descripciones no se considera dentro de los objetivos del proyecto, si bien se analizarán ciertos aspectos (e.g., tamaño de los niveles a transmitir) en la sección de resultados.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Capítulo 1**. Introducción, objetivos y motivación del proyecto.
- **Capítulo 2**. Breve análisis del estándar MPEG-7 y las tecnologías utilizadas para la implementación del sistema.
- **Capítulo 3**. Descripción del diseño y arquitectura del sistema. Además también se analiza el formato de los datos a usar por dicho sistema.
- **Capítulo 4**. Descripción del desarrollo e implementación del diseño del sistema. Problemas en el desarrollo del sistema.
- **Capítulo 5**. Resultados obtenidos con las pruebas realizadas al sistema completo que se ha implementado en el apartado anterior.
- **Capítulo 6**. Conclusiones obtenidas tras el desarrollo del sistema. Relación de posibles líneas futuras de desarrollo y mejoras del sistema.
- **Anexo A** Breve manual con instrucciones de instalación del software utilizado por el sistema.
- **Anexo B** Breve descripción sobre el estándar de codificación BitMap (BMP)
- **Anexo C** Breve descripción del descriptor *VisualDescriptor DS*
- **Anexo D** Breve descripción del estándar MPEG-7.
- **Anexo E** Breve descripción del estándar eXtensive Markup Language**.**
- **Anexo F** Apartado en el que se explican diversas cuestiones matemáticas sobre las funciones gamma y gamma incompletas
- **Anexo G** Ejemplos de codificación en MPEG-7 de una imagen y de una “Moving Region” de la secuencia Hall Monitor.

2 Estado del arte

2.1 Análisis de secuencias de vídeo-seguridad

2.1.1 Introducción

Los sistemas de vídeo-seguridad[1] tienen cada vez mayor demanda, especialmente, para garantizar la seguridad de interiores y alrededores de edificios. Gracias a la evolución tecnológica se ha logrado que la instalación de cámaras de vídeo no precise de altas inversiones económicas y, por tanto, la mayoría de bancos, estaciones, aeropuertos, etc., incorporan en sus instalaciones algún sistema, más o menos complejo, de seguridad basado en vídeo-seguridad.

Históricamente, las tareas de interpretación de vídeo de alto nivel relacionadas con la seguridad (vigilancia) son llevadas a cabo, en su totalidad, por un operario que tiene que procesar una gran cantidad de información de vídeo que se le muestra en uno o varios monitores.

A su vez, es bien sabido, que el operario vigila varios escenarios simultáneamente y realiza este trabajo durante varias horas seguidas que pueden llevar a que su atención disminuya y, por tanto, la probabilidad de pasar por alto situaciones potencialmente peligrosas.

Figura 2-1: Sala de control de cámaras de vídeo-seguridad

[1] Del inglés surveillance. Con este término nos referimos a aquellos sistemas de seguridad que hacen uso de secuencias de vídeo. También se usa el término vídeo vigilancia (vídeo-vigilancia): en este texto cuando nos refiramos a dicho concepto usaremos el término vídeo-seguridad, al considerarlo más amplio.

Para conseguir sistemas de seguridad más robustos basados en vídeo vigilancia, al operario humano se le puede ayudar proporcionándole herramientas automáticas de procesado, sistemas capaces de llamar su atención cuando se produce una situación peligrosa o extraña en el entorno que vigila y que permitan recuperar fácilmente la parte de la secuencia de vídeo relativa al evento que representa la razón por la que se le ha alertado.

Para poder implementar dichas herramientas automáticas de procesado, se necesita previamente un análisis de la secuencia de vídeo-seguridad. Mediante este análisis lo que buscaremos será separar las secuencias en objetos móviles y estáticos (*foreground*) y fondo (*background*).

Las principales dificultades que se pueden encontrar en este tipo de análisis son las siguientes:

- Imágenes del mundo real: muchos detectores de movimiento están adaptados a ciertas características de un medio (intensidad luminosa, calida de la cámara,…) y tienen una fuerte dependencia. Con lo cual ante imágenes reales, que sufren grandes cambios de dichas características, los detectores de movimiento se comportan de manera distinta.

- Sistemas de uso fácil y sencillo: todo aquel sistema de vídeo-seguridad ha de ser fácil de instalar y de manejar.

- Operación durante largos periodos de tiempo: en etapas de larga de duración, las condiciones para las que fue configurado el detector probablemente cambiarán (salvo en medios muy controlados) con lo cual nuestro sistema se podrá comportar de manera distinta. La solución es que el sistema se adapte progresivamente para ir ajustándose a los cambios.

- Coste computacional bajo: el análisis de las secuencias de vídeo-seguridad se ha realizar en un tiempo razonable (lo más próximo a tiempo real) debido a la naturaleza de la detección de eventos.

Relacionados con la vídeo-seguridad, nos centraremos en tres aspectos principales en la detección y seguimiento de objetos. Estos son:

- Separación fondo-objeto en secuencias de vídeo-seguridad
- Seguimiento de objetos en secuencias de vídeo-seguridad
- Generación de fondo en secuencias de vídeo-seguridad

2.1.2 Separación fondo-objeto en secuencias de vídeo-seguridad

El primer paso en cualquier proceso de análisis de imágenes es la segmentación [4]. Mediante la segmentación vamos a dividir la imagen en las partes u objetos que la forman. El nivel al que se realiza esta subdivisión depende de la aplicación en particular, es decir, la segmentación terminará cuando se hayan detectado todos los objetos de interés para la aplicación. En general la segmentación automática es una de las tareas más complicadas dentro del procesado de imagen.

Los algoritmos de segmentación[2] de imágenes generalmente se basan en dos propiedades básicas de los niveles de gris (o su luminancia calculada a partir de las componentes de color) de la imagen: discontinuidad y similitud. Dentro de la primera categoría se intenta dividir la imagen basándose en los cambios bruscos en el nivel de gris. Las áreas de interés de esta categoría son la detección de puntos, líneas y bordes en la imagen. Las áreas dentro de la segunda categoría están basadas en las técnicas de umbrales, crecimiento de regiones, y técnicas de división y fusión.

En la actualidad existen muchas técnicas de segmentación de objetos en una secuencia de imágenes relacionada con la vídeo-seguridad. Diversos investigadores han dedicado su tiempo a dar con la solución de este problema. En este apartado se comentan las más conocidas de ellas:

- **Segmentación de objetos móviles basándose en el ruido introducido por la cámara**

En el articulo ***"Semantic Video Analysis for Adaptive Content Delivery and Automatic Description***"[5] se desarrolla una posible solución al problema de la segmentación de objetos en secuencias de video-seguridad.

En este artículo se diseña un sistema que antes de codificar un determinado video, realiza un análisis de él para así poder realizar una codificación semántica. Con este análisis previo de la secuencia se pretende diferenciar o extraer zonas relevantes de la secuencia. Estas zonas se podrán codificar con menor o mayor calidad o codificarse de manera textual. El sistema es el siguiente:

[2]Para más información sobre algoritmos de segmentación véase la siguiente página Web http://iris.usc.edu/Vision-Notes/bibliography/contentssegment.html#2-D%20Region%20Segmentation%20Techniques,%20Snakes

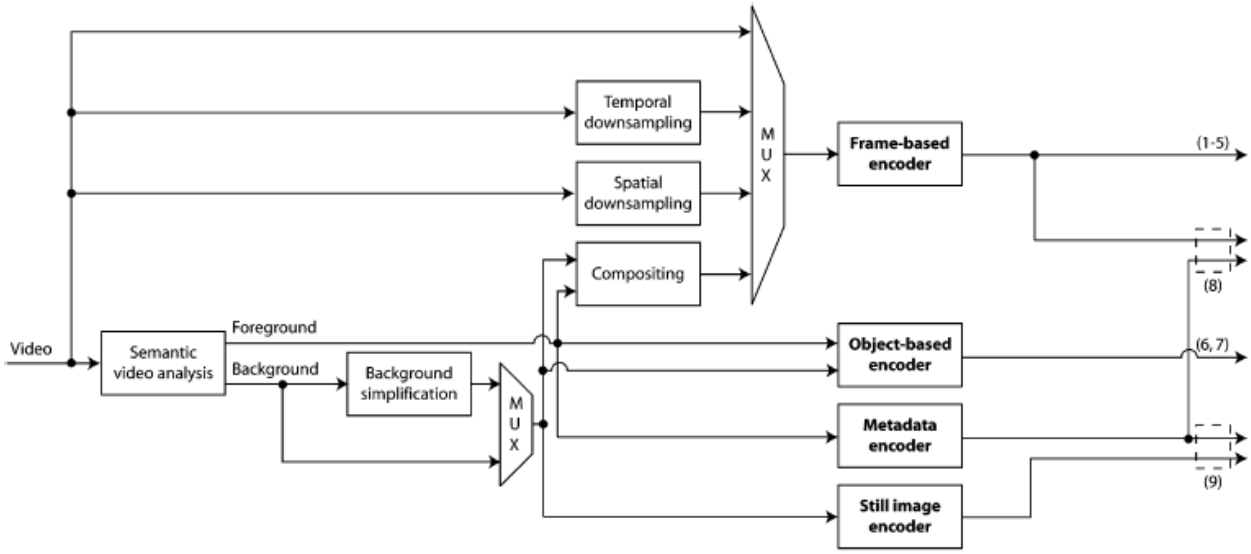


Figura 2-2: Sistema de codificación de vídeo basado en análisis semántico y descripciones (metadatos) [5]

donde para este estudio del arte lo que nos interesa es el módulo "*Semantic Video Análisis*". En dicho módulo es donde se realiza la separación entre objetos y fondo para la posterior codificación semántica.

En el artículo se nos comenta que para la extracción de objetos en movimiento, la información de movimiento puede ser usada como semántica.

Una herramienta típica para realizar la segmentación es la detección de movimiento. Se emplean distintos algoritmos para condiciones donde las cámaras estén en movimiento o estáticas. Nos centraremos en el caso de cámaras estáticas ya que en este PFC solo se analizará ese caso.

Concretamente lo que nos interesa es el detector de movimiento desarrollado en el artículo. Este decidirá si cada píxel de la imagen corresponde al fondo o a un objeto en movimiento. La decisión se toma mediante una umbralización de la diferencia del cuadro[3] actual y del cuadro que representa el fondo de la secuencia. El cuadro que representa el fondo (background) es generado dinámicamente basándose en otro algoritmo que usa información temporal. La umbralización que se plantea en este artículo permite reducir el efecto del ruido de la cámara. El umbral que se usa para el detector de movimiento deberá ser adaptado a las características de la cámara y del lugar donde se este grabando la secuencia.

Con todos estos supuestos mencionados anteriormente, lo que se quiere es determinar si la diferencia entre un píxel del cuadro actual y el mismo píxel del cuadro anterior es debida al ruido

[3] Cuando utilizamos este término en este documento, nos estaremos refiriendo a los frames que componen una secuencia de vídeo.

de la cámara (en caso de que no sea así debería ser detectado como movimiento el píxel) o debida a que hay movimiento.

Inicialmente se supone que no existe ningún objeto moviéndose en el cuadro diferencia comentado anteriormente (hipótesis H_0) y que el cambio de valor de píxel es debido al ruido de la cámara (considerado aditivo y que sigue una distribución gaussiana con varianza σ). Así con la hipótesis Ho, la función de densidad de probabilidad condicional del cuadro diferencia sigue una distribución χ_q^2 con q grados de libertad (donde q es el tamaño de la ventana alrededor del píxel que estamos analizando) y esta definida por:

$$f(g(i,j)|H_0) = \frac{1}{2^{q/2}\sigma^q \Gamma(q/2)} g(i,j)^{(q-2)/2} e^{-g(i,j)^2/2\sigma^2}$$

donde Γ(x) es la función gamma y σ es la varianza del ruido.

A partir de esta función de densidad de probabilidad se puede derivar en el siguiente test:

$$P\{g(i,j) \geq \tau(i,j)|H_0\} = \frac{\Gamma\left(\frac{q}{2}, \frac{g(i,j)^2}{2\sigma^2}\right)}{\Gamma\left(\frac{q}{2}\right)}$$

en el que se establece un determinado umbral (determinado empíricamente con valores desde 10^{-2} hasta 10^{-6}). Estableceremos que si el valor de la probabilidad es menor que ese umbral, el píxel no satisfará la condición Ho y entonces pertenecerá a un objeto en movimiento. En caso contrario el píxel pertenecerá al fondo.

Este método expuesto es bastante sencillo de aplicar pero tiene el problema de que es bastante dependiente de la secuencia de imágenes y del ruido que introduce la cámara que queramos analizar.

En el artículo "***Robust Motion Detector for Video Surveillance Applications***"[6] se presenta otra posible solución para el diseño de un sistema automático de detección de movimiento. El objetivo del sensor de video[6] es trabajar para aplicaciones de video-seguridad durante un largo periodo de tiempo y con cambios en el nivel de ruido de las secuencias captadas por la cámara.

En este artículo se nos introduce el concepto de sensor de vídeo, que no es más que una herramienta de análisis de video digital que extrae información de interés de una determinada secuencia.

Se distinguirán dos tipos de variaciones en la secuencia: las debidas al efecto del ruido y las debidas al movimiento de objetos. Para distinguir de qué tipo es cada variación, se aplicará un umbral a una medida de similitud entre cuadros consecutivos. A dicha medida de similitud le sumaremos el efecto del ruido (caracterizado por su varianza) y así calcularemos el umbral para iniciar la detección.

En este artículo también se hace especial hincapié en la actualización de la varianza del ruido, que se hace necesario si queremos que nuestro sistema responda coherentemente ante variaciones de las condiciones del entorno (e.g, iluminación, temperatura). Una posible estrategia es la de actualizar dicho parámetro cuando no se detecte movimiento en la secuencia.

Respecto a los temas de actualización de parámetros, también se considera la actualización del umbral de detección de movimiento; para ello la mejor estrategia es actualizarlo cuando sabemos (a priori) que en dicha secuencia no se va a detectar ningún movimiento.

- **Segmentación de objetos móviles basándose en la detección de contornos**

En el artículo "***Automatic vehicle image extraction based on spatio-temporal image analysis***"[7] se diseña e implementa un sistema para identificar vehículos en secuencias de videocámaras que están filmando tramos de autopistas.

La detección de movimiento se lleva a cabo aplicando la *Transformada Hough*[4], un algoritmo orientado a detectar contornos con formas geométricas determinadas (rectas, circunferencias, etc.). La detección de rectas se basa en la ecuación de la recta en el plano. Por cada punto de gradiente de la imagen aumentamos el valor de un punto en un espacio equivalente de parámetros de una recta. De este modo, las zonas en el nuevo espacio de alta intensidad corresponden a rectas presentes en la imagen original. De esta manera podemos detectar cualquier forma describible con una ecuación.

[4] http://en.wikipedia.org/wiki/Hough_transform

2.1.3 Seguimiento de objetos en secuencias de vídeo-seguridad

El seguimiento de objetos en tiempo real es una tarea crítica en muchas aplicaciones de vídeo-seguridad[1], asistencia al conductor[9], interfaces de usuario perceptuales[10], compresión de video basada en objetos[11],…

En un contexto de percepción visual, el seguimiento de un objeto es, en esencia, un proceso en el que se efectúa la detección del objeto u objetos móviles y su "persecución" a través de las secuencias de imágenes del entorno adquiridas por una o varias cámaras (estáticas o móviles). El proceso de seguimiento puede implicar a cualquier objeto móvil presente en la escena, sin reconocer de qué objeto se trata (seguimiento precategórico[19]) o el seguimiento de uno o varios objetos específicos del cual se tiene un modelo predefinido (seguimiento categórico[19]).

Tradicionalmente existen dos aproximaciones para resolver el problema del seguimiento:

- *Las aproximaciones basadas en el análisis del contenido de los píxeles*. Estas técnicas determinan las zonas de la imagen donde se produce movimiento y realizan el seguimiento mediante un análisis del contenido de los píxeles. Ejemplos de estas dos aproximaciones son los métodos basados en el análisis de movimientos diferenciales[12] y las técnicas de flujo óptico[13].

- *Las aproximaciones basadas en modelos*. Estas técnicas implican localizar, en cada una de las imágenes que componen la secuencia, la posición del objeto móvil del que se dispone de un modelo. Los modelos empleados suelen construirse "a mano", inducirse a partir de una secuencia de ejemplos o adquirirse dinámicamente a partir del objeto móvil. Los métodos basados en modelo incluyen: métodos correlacionales[14], métodos basados en correspondencia[15], métodos de filtrado[16], métodos basados en contornos deformables activos [17] y métodos basados en filtrado predictivo [18].

Siguiendo estas técnicas, se han diseñado diversos algoritmos de seguimiento que después se han implementado en los múltiples sistemas. Ahora se procede a comentar algunas de las soluciones aplicadas en la actualidad.

En el artículo **"Seguimiento de objetos Móviles usando la Distancia de Hausdorff"**[19] se propone una solución al seguimiento de objetos según un esquema basado en modelos. El núcleo de la solución adoptada lo constituye un proceso de comparación con el modelo planteado como un problema de búsqueda y efectuando la utilización de una medida de comparación basada en la

distancia de Hausdorff[20]. El método de seguimiento planteado esta basado íntegramente en la comparación de estructuras geométricas 2D entre escenas consecutivas. La distancia de Hausdorff es una medida de la distancia definida entre dos conjuntos de puntos. Formalmente, la distancia definida entre los conjuntos A y B, donde A y B son conjuntos de puntos (p.e. puntos de la imagen, donde A representa la imagen y B representa al modelo)

$$H(A,B) = \max(h(A,B), h(B,A))$$

donde

$$h(A,B) = \max_{a \in A} \min_{b \in B} \|a - b\|$$

y $\|a-b\|$ es la distancia entre los puntos a y b medidos por alguna norma (L_1, L_2,…). La función $h(A,B)$ se conoce como la distancia directa de Hausdorff desde A a B, y ordena cada punto de A basado en su distancia al punto más cercano de B.

En el artículo **"Detecting and Tracking Moving Objects for Video Surveillance"**[20], el método de seguimiento basado en el análisis de los contenidos de los píxeles. Para ello se define un pequeño conjunto de puntos característicos y se les aplica una transformación. Posteriormente se pueden incorporar a la ecuación de flujo óptico para realizar la estimación y seguimiento del movimiento. Adicionalmente también se incorporan técnicas correlacionales para mejorara el seguimiento de los objetos a través de la secuencia de imágenes.

2.1.4 Generación de fondo en secuencias de vídeo-seguridad

La cuestión de la generación de fondo es otro problema relacionado con el apartado anterior: no se podrá realizar un estimador de movimiento robusto sino tenemos un buen algoritmo de generación de fondo. Ocurriría un grave fallo en nuestro sistema si el fondo de nuestra secuencia de imágenes fuese cambiando con el paso del tiempo y nuestro algoritmo fuese incapaz de seguir ese cambio. Llegaría un momento en el que ya no se parecería en nada el fondo que nuestro sistema estima con el que existe en realidad. Este hecho provocaría múltiples fallos en la detección.

Los principales problemas a los que nos enfrentamos para la detección del fondo son los siguientes:

- Cambios de iluminación (graduales y repentinos)
- Cambios de movimiento (oscilaciones de la cámara, movimiento de objetos con alta frecuencia,…)

- Cambios en la geometría del fondo (por ejemplo coches aparcados)

Para intentar solucionar estos problemas se han desarrollado una gran cantidad de algoritmos y así luego poder realizar una estimación correcta de movimiento en la secuencia analizada. Se han investigado diversas técnicas que se exponen a continuación:

- Métodos básicos
    - El fondo estimado es el cuadro anterior
        - Este cálculo de fondo es usado en métodos como Cuadro diferencia. Este método solo funciona bajo determinadas condiciones de velocidad de objetos y velocidad de cuadro (fps). La detección es muy dependiente del umbral (*Th*) que seleccionemos.

$$\left| frame_i - frame_{i-1} \right| > Th$$

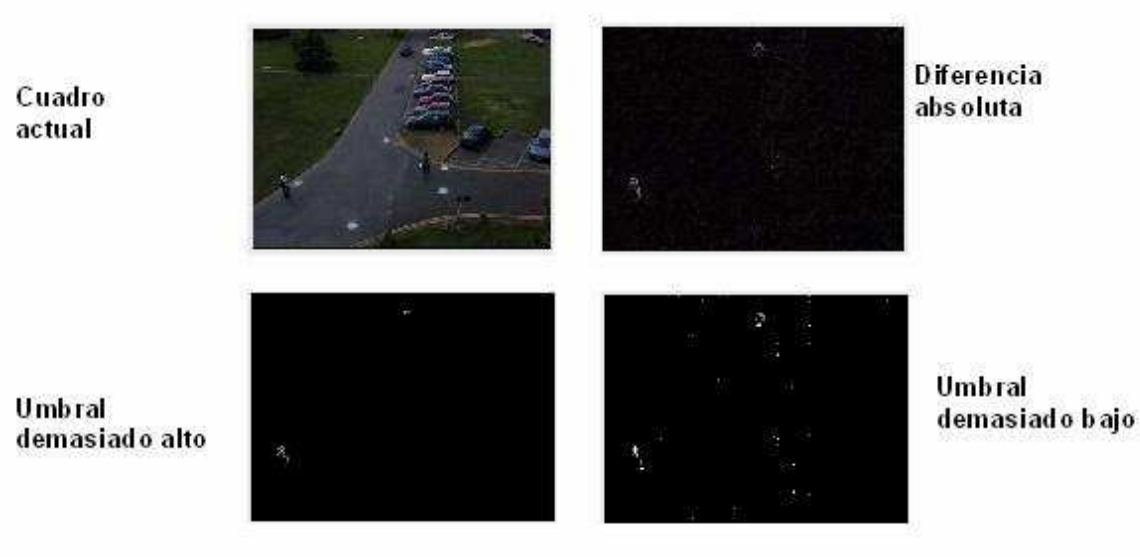


Figura 2-3: Efecto del umbral para la estimación de fondo en el método *cuadro diferencia* [20]

    - Fondo calculado como media o mediana de los previos 'n' cuadros en la secuencia (método *average o median*) [22]
        - El problema de esta estimación es la consecuente inclusión de objetos en movimiento que pasan a formar parte del background (con el consiguiente error en posteriores detecciones). Otro problema seria el requerimiento de memoria, pues tendremos que almacenar los 'n' cuadros anteriores

- Actualización progresiva de fondo (método *running average*) [23]

    - Mejora del método *average* evitando el uso de memoria. Sigue la siguiente fórmula:

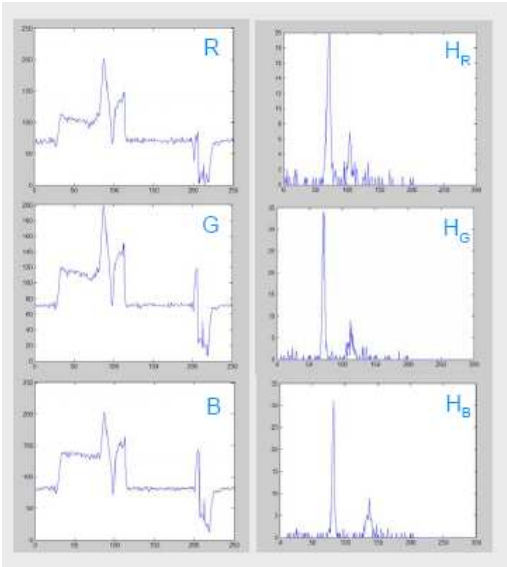
$$B_{i+1}(i, j) = \alpha F_i(i, j) + (1 - \alpha) B_i(i, j)$$

donde típicamente el factor de actualización (α) vale 0.05

- Generación de fondo basada en histogramas

    - Para decidir si un píxel pertenece al fondo nos fijaremos en el histograma de dicho píxel a través de los últimos 'n' cuadros. En la siguiente figura se muestra el histograma de cada banda (RGB) del píxel a analizar. Realizando una observación del histograma podremos determinar si el nuevo cambio del valor del píxel es debido a un posible movimiento de un objeto o a otros factores (ruido de la imagen, cambio de luminosidad…)

Píxel a analizar



Histograma de dicho píxel (RGB)

Figura 2-4: Calculo de fondo mediante la técnica de histogramas

- Método de selectividad

    - Para cada nuevo cuadro, los píxeles son clasificados como fondo (*background*) o primer plano (*foreground*). Si son clasificados como

primer plano, entonces no se toman en cuenta para la actualización del fondo.

$B_{i+1}(i, j) = \alpha F_i(i, j) + (1-\alpha) B_i(i, j)$ si $F_i(i, j)$ es fondo

$B_{i+1}(i, j) = B_i(i, j)$ si $F_i(i, j)$ es primer plano

- Métodos avanzados
  - Método "Running Gaussian Average"[24]
    - Se intenta ajustar una distribución gaussiana (μ, σ) sobre el histograma: así obtenemos la función de densidad de probabilidad del fondo.
  - Método "Mixture of Gaussianas"[25]
    - Basándonos en el método anterior, consideramos un ajuste a una mezcla de gaussianas ponderadas (μ_i, σ_i, w_i). Con este modelo podremos hacer frente a distribuciones multimodales de fondos (backgrounds).
  - Método "Kernel Density Estimators"[26]
    - La función de densidad de probabilidad (fdp) del fondo la obtenemos a través del histograma de los "n" últimos valores,, cada uno alisado con un núcleo gaussiano.
    - Si la fdp(x) > Th, entonces el píxel x es clasificado como background
    - Este método implica selectividad y grandes requerimientos de memoria (n * size(cuadro))
  - Método Mean Shift Based Stimation[27]
    - El método detecta los modos de una distribución multimodal, para ello usa un gradiente ascendiente y la matriz de covarianzas.
    - Es un método iterativo.
    - El problema de la implementación estándar es que es demasiado lenta.

- Método “EigenBackgrounds”[28]

    - Mediante el análisis de los principales componentes (PCA, Principal Component Analysis) podemos reducir la dimensionalidad de un espacio

    - Dicho análisis puede ser aplicado a una secuencia de “n” cuadros para computar los autofondos (eigenbackgrounds)

    - Todos los cuadros son computados una sola vez y para ello se usan conceptos como matriz de covarianzas, autovalores (eigenvalores), autovectores (eigenvectores),…

Como resumen final de los métodos anteriormente expuestos:

Método	**Velocidad**	**Requerimientos de memoria**
Average	Rápido	Alto
Running Average	Rápido	Bajo
Median	Rápido	Alto
Mean-Shift Estándar	Lento	Alto
Mean-Shift Optimizado	Intermedio	Alto
Mixture Of Gaussians	Intermedio	Intermedio
KDE	Intermedio	Alto
Eigenbackgrounds	Intermedio	Intermedio

Tabla 2-1. Comparación de métodos de generación y actualización de background

Hay que destacar que la precisión con la que se obtienen los resultados es muy dependiente de la secuencia de imágenes que se este analizando. De todas maneras parece ser los métodos KDE y mean-shift son los que obtienen mejores resultados.

No todas las técnicas de estimación de fondo han sido implementadas en los sistemas de análisis de movimiento y seguimiento de objetos. En [7] y [8] se muestran las algunas de las técnicas más conocidas que se han implementado en diversos sistemas.

2.2 Generación de descripciones

2.2.1 Introducción

La cantidad de contenido digital multimedia disponible para los consumidores está creciendo constantemente debido a la existencia de una enorme cantidad de fuentes de contenidos digitales (cámaras digitales, teléfonos móviles,…). Con este incremento del contenido, es muy importante para los usuarios que sea posible la búsqueda y navegación por dicho contenido de una manera rápida y eficiente. Descripciones y anotaciones del contenido se han hecho necesarios para poder permitir una navegación, búsqueda y filtrado por dicho contenido.

Debido a la gran cantidad de contenido disponible, dicha tarea sería imposible de realizar manualmente. El objetivo es tener un sistema automático que clasifique y haga las anotaciones necesarias para poder realizar una navegación, búsqueda y filtrado eficientes.

En el desarrollo de este PFC, nos interesa la generación de descripciones referentes a imágenes estáticas y a objetos en movimiento (y su posterior seguimiento).

2.2.2 Sistemas actuales

Actualmente se están desarrollando numerosos sistemas automáticos que generan descriptores del contenido multimedia que se almacena.

El estándar para las descripciones más usado en la actualidad es MPEG-7. Numerosos sistemas se han desarrollado basándose en dicho estándar. Ahora se procederá a comentar algunos sistemas:

Por ejemplo, en "***Automatic Generation of MPEG-7 Compliant XML Document for Motion Trajectory Descriptor in Sports Video***"[29] se desarrolla un sistema completo para extracción de descriptores en MPEG-7. Este sistema analiza diversas características de videos sobre deportes y después escribe en MPEG-7 dichas características (color, movimiento de cámara, …)

Para describir el movimiento de los objetos, se usar el descriptor *MotionTrayectory* . Para conseguir todos los datos referentes al movimiento de los objetos se necesita hacer una estimación del movimiento de la cámara. Para ello se usa CameraMotion (que utiliza los Vectores de Movimiento de MPEG2[5]).

[5]Para más información general sobre MPEG-2 y vectores de movimiento (VM) consultar la página Web http://es.wikipedia.org/wiki/MPEG-2

2.3 MPEG-7: estándar Descripción de contenido multimedia

2.3.1 Introducción

Hace años el acceso a contenidos multimedia solía ser una tarea que no entrañaba ningún problema. Esto era debido a la manera de acceder a ellos y que existía disponible poco contenido multimedia. No existía una necesidad de clasificar el contenido multimedia

Actualmente una enorme y creciente cantidad de contenido audiovisual se ha puesto disponible a todo el mundo gracias a la revolución digital y se ha difundido principalmente a través de internet. En esta nueva situación, el valor de la información empieza a depender de cómo es el acceso a ella. Es decir tenemos un acceso complejo debido a la gran cantidad de información y fuentes que la distribuyen. Así pues el valor de la información ya no dependerá solamente de ella misma, sino también de la facilidad para ser encontrada, recuperada y filtrada.

MPEG-7 trata de ser la solución a esta necesidad. Es un estándar internacional de la ISO/IEC, desarrollado por el grupo MPEG (Moving Picture Experts Group), que ha desarrollado otros estándares tan importantes como son el MPEG-1, MPEG-2 y MPEG-4. MPEG-4 es el primer estándar real de representación multimedia, permitiendo interactividad y la combinación de material natural y sintético, codificado como objetos.

El estándar MPEG-7, conocido como *Interfaz de Descripción de Contenido Multimedia*, proporciona un amplio conjunto de herramientas estándar para describir contenido multimedia, tanto para usuarios humanos como para sistemas automáticos que procesen información audiovisual. Estas herramientas de descripción (conocidas como Description Tools, que son los elementos de metadatos y su estructura y relaciones, definidas en el estándar como Descriptores (D) y Esquemas de Descripción (DS)) sirven para crear descripciones que serán la base para aplicaciones que permitan este necesario acceso eficiente a contenido multimedia. Es una tarea complicada dado el amplio espectro de requisitos y aplicaciones multimedia que pretende abarcar, así como el gran número de características audiovisuales de importancia en este contexto.

En posteriores apartados se procederá a explicar conceptos del estándar referentes al PFC desarrollado: concretamente nos centraremos en la parte 5 "*Multimedia content description interface – Part 5: Multimedia Description Schemes"*.

Para una visión más general del estándar véase el Anexo **"MPEG-7: ESTÁNDAR DESCRIPCIÓN DE CONTENIDO MULTIMEDIA"**

2.3.2 Contexto de aplicación

La aplicación del estándar en nuestro PFC tendrá un enfoque claro, el análisis de secuencias de video-seguridad, la generación de descriptores de imágenes estáticas y de descriptores de movimiento a partir de las secuencias de imágenes que recibimos de las cámaras.

Para los descriptores de contenido visuales tenemos las siguientes opciones:

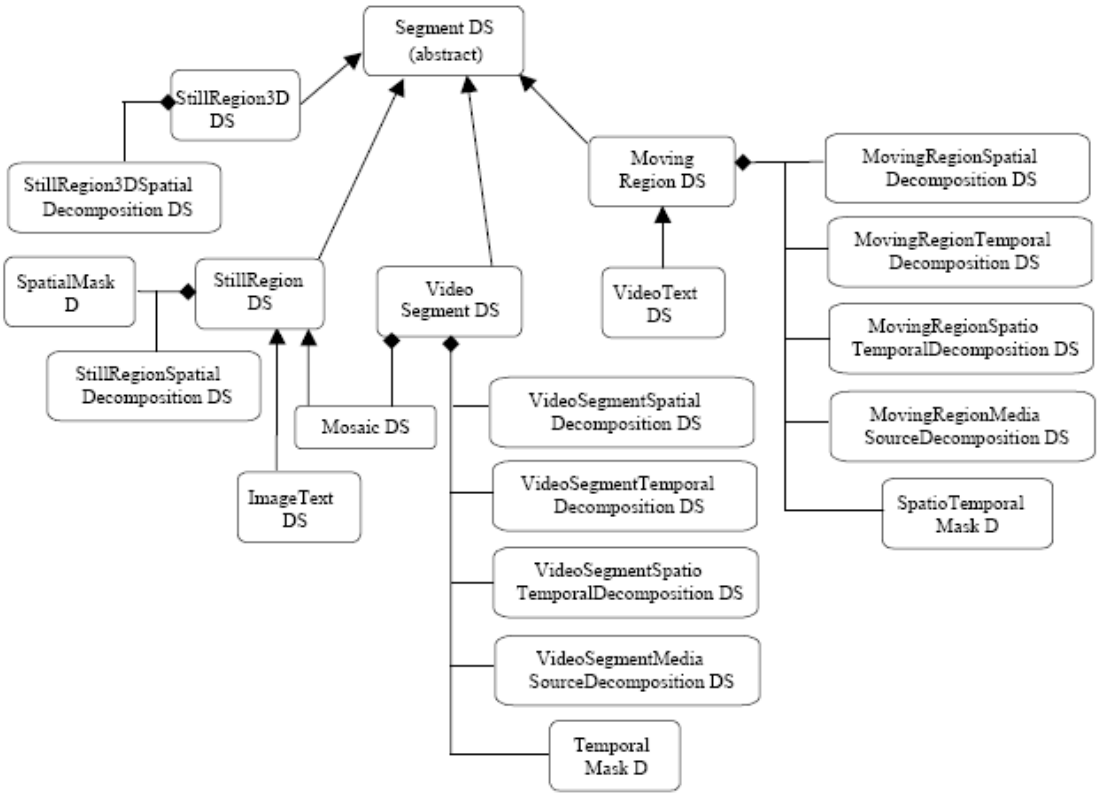


Figura 2-5: Esquemas para la descripción de segmentos de contenido visual

Para ello el estándar nos proporciona unos descriptores Descriptor Schemes[6] *Still Region DS* y *Moving Region DS*. Que procedemos a explicar en los siguientes apartados.

El esquema *Still Region DS* se usara para generar la descripción correspondiente a una imagen estática y el descriptor *Moving Region DS* se usara para generar el descriptor correspondiente a la trayectoria de un objeto.

[6] Esquemas de descripción de contenido multimedia

Adicionalmente a los descriptores que se han mencionado, se necesitara incluir alguna información de color sobre las imágenes. Para ello usaremos el esquema *VisualDescriptor DS.* Dicho esquema se encuentra en la parte 3 del estándar "*Multimedia content description interface – Part 3: Visual*"

MPEG-7 es un estándar de descripción de contenido multimedia que soportará estos requisitos operacionales. Estos requisitos se aplican tanto para aplicaciones de tiempo real y de no tiempo real. MPEG-7 no estandariza ni evalúa aplicaciones.

2.3.3 Still Region DS

2.3.3.1 Introducción

Como se comentó en el contexto de la aplicación, para generar un descriptor correspondiente a una imagen estática usaremos el esquema *Still Region DS.*

El esquema *Still Region DS* describe una imagen o una región 2D de una imagen o un cuadro de un video. El esquema *Still Region DS* extiende *Segment DS*. En el caso de imágenes digitales, una "S*till Region*" puede corresponder a un solo píxel, un grupo arbitrario de píxeles o la imagen entera. Dicha región no tiene porque estar conectada en el espacio. El descriptor *SpatialMask DS* es opcional y puede describir una mascara en dos dimensiones que nos indique el conjunto de sub-regiones no conectadas que puede contener la imagen ("*Still Region*").

2.3.3.2 *Sintaxis Still Region*

La sintaxis de una descripción de una región estática es la siguiente

```
<!-- ######################################## -->
<!-- Definition of StillRegion DS (11.4.2) -->
<!-- ######################################## -->

<!-- Definition of StillRegion DS -->
<complexType name="StillRegionType">
      <complexContent>
            <extension base="mpeg7:SegmentType">
                  <sequence>
                        <choice minOccurs="0">
                              <element name="SpatialLocator"
                                          type="mpeg7:RegionLocatorType"/>
                              <element name="SpatialMask"
                                          type="mpeg7:SpatialMaskType"/>
                        </choice>
                        <choice minOccurs="0">
                              <element name="MediaTimePoint"
                                          type="mpeg7:mediaTimePointType"/>
                              <element name="MediaRelTimePoint"
                                          type="mpeg7:MediaRelTimePointType"/>
                              <element name="MediaRelIncrTimePoint"
                                          type="mpeg7:MediaRelIncrTimePointType"/>
                        </choice>
                        <choice minOccurs="0" maxOccurs="unbounded">
                              <element name="VisualDescriptor" type="mpeg7:VisualDType"/>
                              <element name="VisualDescriptionScheme"
                                          type="mpeg7:VisualDSType"/>
                              <element name="GridLayoutDescriptors"
                                          type="mpeg7:GridLayoutType"/>
                        </choice>
                              <element name="MultipleView" type="mpeg7:MultipleViewType"
                                          minOccurs="0"/>
                              <element name="SpatialDecomposition"
                                          type="mpeg7:StillRegionSpatialDecompositionType
                                    "
                                          minOccurs="0" maxOccurs="unbounded"/>
                  </sequence>
            </extension>
      </complexContent>
</complexType>
```

Figura 2-6: Definición de Still Region DS (ISO/IEC 15938-5)

Ahora procedemos a describir cada elemento de la sintaxis de un descriptor de una región estática.

- *StillRegionType* → Describe una imagen o una región espacial 2D de una imagen o un cuadro de un vídeo. La región no tiene porque estar conectada en el espacio. El esquema *StillRegion DS* usa las herramientas de descriptores visuales (*VisualDType* y *VisualDSType*) para describir las características visuales de las regiones estáticas. La localización espacial o composición de la región estática se puede describir opcionalmente usando la elección de *SpatialLocator* o *SpatialMask*. *StillRegionType* extiende el esquema *SegmentType.*

- *SpatialLocator* → Describe la localización espacial de una región estática usando un *RegionLocatorType* (definido en ISO/IEC 15938-3). El localizador espacial describe sub-regiones conectadas en el espacio que forman la región estática.

- *SpatialMask* → Describe una mascara espacial 2-D que define la composición de la región estática. Dicha región esta formada por un conjunto de sub.-regiones descritas por *SpatialMask*. Si no existiera, entonces la región estaría compuesta solo por una región conectada definida por *SpatialLocator*.

- *MediaTimePoint* → Indica el punto temporal de la región estática proveniente de un video usando *mediaTimePointType* (opcional)

- *MediaRelTimePoint* → Indica el punto temporal de la región estática proveniente de un video usando *MediaRelTimePointType* (opcional)

- *MediaRelIncrTimePoint* → Indica el punto temporal de la región estática proveniente de un video usando *MediaRelIncrTimePointType* (opcional)

- *VisualDescriptor* → Describe una característica visual de una región estática usando un descriptor visual (opcional). Para ello se utilizan los esquemas *VisualDTypes* definidos en ISO/IEC 15938-3.

- *VisualDescriptionScheme* → Describe una característica visual de una región estática usando un DS descriptor visual (opcional). Para ello se utilizan los esquemas *VisualDSTypes* definidos en ISO/IEC 15938-3.

2.3.4 Moving Region DS

2.3.4.1 Introducción

El esquema *MovingRegion DS* describe un vídeo o una región espacio-temporal de un vídeo. *MovingRegion DS* extiende *Segment DS*. En el caso de video digital, una región de movimiento puede corresponder a un grupo arbitrario de píxeles o el video completo.

La región de movimiento no tiene porque estar conectada temporalmente o espacialmente. El esquema *SpatioTemporalMask DS* es opcional y describe una máscara espacio-temporal dando un conjunto de sub-regiones no conectadas en el caso de que la región de movimiento este formada por sub-regiones no conectadas.

2.3.4.2 Sintaxis Moving Region

```
<!-- ######################################### -->
<!-- Definition of MovingRegion DS (11.4.10) -->
<!-- ######################################### -->

<!-- Definition of MovingRegion DS -->
<complexType name="MovingRegionType">
       <complexContent>
              <extension base="mpeg7:SegmentType">
                     <sequence>
                            <choice minOccurs="0">
                                   <element name="SpatioTemporalLocator"
                                          type="mpeg7:SpatioTemporalLocatorType"/>
                                   <element name="SpatioTemporalMask"
                                          type="mpeg7:SpatioTemporalMaskType"/>
                            </choice>
                            <choice minOccurs="0" maxOccurs="unbounded">
                                   <element name="VisualDescriptor" type="mpeg7:VisualDType"/>
                                   <element name="VisualDescriptionScheme" type="mpeg7:VisualDSType"/>
                                   <element name="VisualTimeSeriesDescriptor"
                                          type="mpeg7:VisualTimeSeriesType"/>
                            </choice>
                            <element name="MultipleView" type="mpeg7:MultipleViewType"
                                          minOccurs="0"/>
                            <choice minOccurs="0" maxOccurs="unbounded">
                                   <element name="SpatialDecomposition"
                                          type="mpeg7:MovingRegionSpatialDecompositionType"/>
                                   <element name="TemporalDecomposition"
                                          type="mpeg7:MovingRegionTemporalDecompositionType"/>
                                   <element name="SpatioTemporalDecomposition"
                                          type="mpeg7:MovingRegionSpatioTemporalDecompositionType"/>
                                   <element name="MediaSourceDecomposition"
                                          type="mpeg7:MovingRegionMediaSourceDecompositionType"/>
                            </choice>
                     </sequence>
              </extension>
       </complexContent>
</complexType>
```

Figura 2-7: Definición de MovingRegion DS (ISO/IEC 15938-5)

Ahora procedemos a describir cada elemento de la sintaxis de un descriptor de una region de movimiento:

- *MovingRegionType* → Describe un vídeo o una región espacio-temporal de un vídeo. La región de movimiento no tiene porque estar conectada en espacio o en tiempo. El esquema *MovingRegion DS* usa herramientas de descripción visual (*VisualDType and VisualDSType*) para describir las características visuales de la región de movimiento. La localización espacio-temporal o composición de la región de movimiento puede ser descrita opcionalmente usando los esquemas SpatioTemporalLocator o SpatioTemporalMask. MovingRegionType extiende SegmentType.

- *SpatioTemporalLocator* → Describe la localización espacio-temporal de la región de movimiento

- *SpatioTemporalLocatorType* (opcional) → El localizador espacio-temporal describe una sub-región conectada espacial o temporalmente que incluye los componentes

espacio-temporales de la región de movimiento. *SpatioTemporalLocatorType* esta definido en ISO/IEC 15938-3.

- *SpatioTemporalMask* (opcional)→ Describe una mascara espacio-temporal que define una composición de la región de movimiento. La región de movimiento esta formada por un conjunto de sub-regiones descritas por el esquema *SpatioTemporalMask*. En el caso de no existir, la región de movimiento esta compuesta por una sola región conectada definida por el esquema *SpatioTemporalLocator*.

- *VisualDescriptor* → Describe una característica visual de una región estática usando un descriptor visual (opcional). Para ello se utilizan los esquemas *VisualDTypes* definidos en ISO/IEC 15938-3.

- *VisualDescriptionScheme* → Describe una característica visual de una región estática usando un esquema de descripción visual (opcional). Para ello se utilizan los esquemas *VisualDSTypes* definidos en ISO/IEC 15938-3.

- *VisualTimeSeriesDescriptor* → Describe una secuencia de características visuales de la región de movimiento (opcional). El esquema *VisualTimeSeriesDescriptor* solamente se aplica en el caso de una región de movimiento conexa. Dicho esquema esta definido en ISO/IEC 15938-3

- *SpatialDecomposition* → Describe la descomposición espacial de una región de movimiento en uno o varios sub-segmentos (opcional).

- *TemporalDecomposition* → Describe la descomposición temporal de una región de movimiento en uno o varios sub-segmentos temporales (opcional).

- *SpatioTemporalDecomposition* → Describe la descomposición espacio-temporal de una región de movimiento en uno o varios sub-segmentos (opcional)

- *MediaSourceDecomposition* → Describe la descomposición de la fuente del contenido de la región de movimiento en uno o varios sub-segmentos (opcional).

2.3.5 Visual Descriptor DS

Dentro del estándar MPEG-7, existe un apartado para realizar descripciones de contenidos visuales[32]. Para realizar la descripción de un contenido visual nos podemos fijar en distintas características, en la figura 2-8 se muestran las principales características que podremos describir:

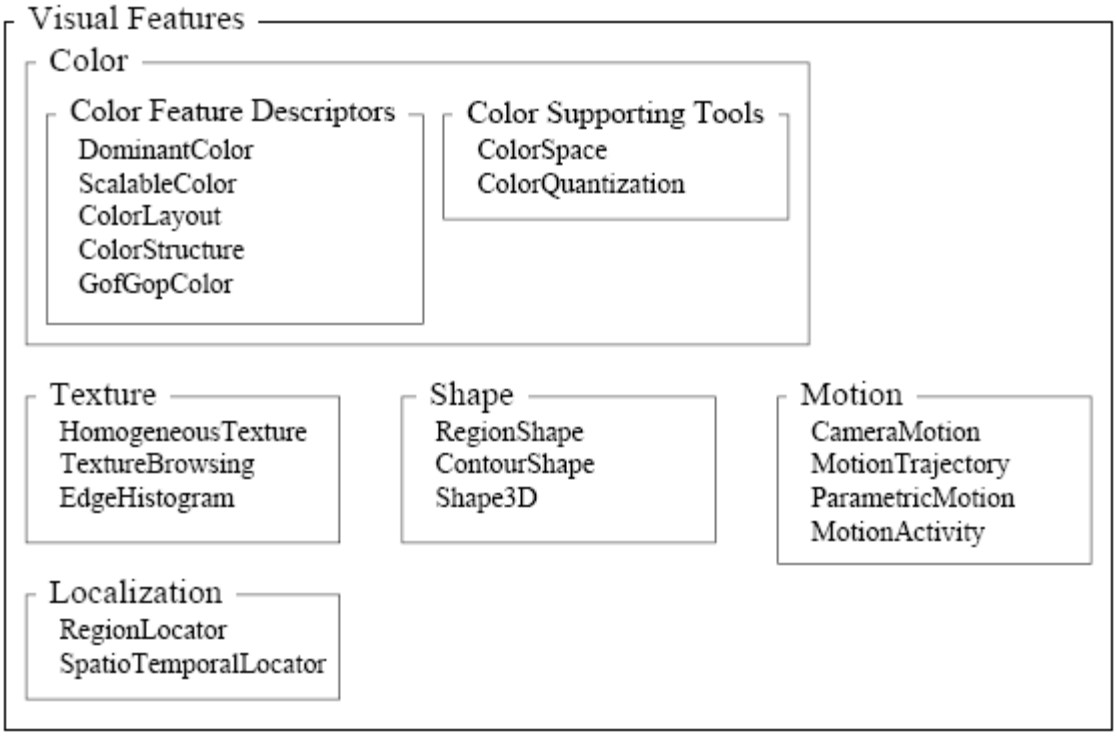


Figura 2-8: Descriptores para representar características visuales (ISO/IEC 15938-3)

En este estudio del arte nos centraremos en cuatro categorías principales (color, textura, forma y movimiento). A continuación se expone de cada categoría las herramientas de descripción existentes:

- Herramientas para la descripción del color (*Color*)
    - *DominantColor DS*. Con esta característica representaremos los colores dominantes de una determinada región.

- *ScalableColor DS*. Con esta característica representaremos la distribución de los distintos colores de una determinada región.
- *ColorLayout DS.* Con este descriptor representaremos la distribución espacial de los colores de una determinada región.
- *ColorStructure DS.* Con este descriptor representaremos la distribución espacial local de una determinada región.

- Herramientas para la descripción de la textura(*Texture*)
  - Los descriptores de textura facilitan la navegación y la recuperación usando dicha característica en bases de datos de imagen y vídeo. Para ello disponemos de las siguientes características: *HomogeneusTexture DS, TextureBrowsing DS y EdgeHistogram DS.*

- Herramientas para la descripción de la forma (*Shape*)
  - *RegionShape DS*. Este descriptor especifica la forma de una región de un objeto.
  - *ContourShape DS*. Este descriptor especifica un contorno cerrado de un objeto 2Do una región en una imagen o una secuencia de vídeo.
  - *Shape3D*. Para la descripción de contornos en 3D

- Herramientas para la descripción del movimiento (*Motion*)
  - *Camera Motion DS*. En este descriptor se especifican un conjunto de operaciones de movimiento básicas con una cámara (*pan y tilt*)
  - *MotionTrayectory DS.* El movimiento de un determinado punto, de un objeto o region, puede ser caracterizado mediante este tipo de descriptor.
  - *ParametricMotion DS*. Este tipo de descriptores caracteriza una evolución de una región arbritaria mediante una transformación geométrica 2D.
  - *MotionActivity DS*. Con este descriptor calcularemos el ritmo del movimiento en una secuencia.

Para más información consultar la parte 3 del estándar ISO/IEC 15938-3[32].

3 Diseño

A lo largo de este capítulo vamos a abordar diferentes aspectos involucrados en el diseño del sistema y las razones por las cuales se ha adoptado una u otra tecnología y se han tomado determinadas decisiones.

Para ello iremos analizando la arquitectura del sistema completo y de sus diferentes bloques funcionales y lo relacionaremos con las tecnologías usadas en cada módulo.

3.1 Introducción

El principal objetivo de este PFC es desarrollar un sistema para dar soporte a la transmisión información relevante de secuencias de cámaras de seguridad a muy baja tasa binaria y adaptable (llegando por supuesto a la posibilidad de transmitir toda la información junto a una descripción asociada). Este contenido será adaptado convenientemente a las características del terminal y recursos red de los que esté haciendo uso, aprovechando al máximo los recursos de que se disponen, y no malgastando recursos si luego no es posible hacer uso de esas características en el terminal cliente; así como a las preferencias del usuario (aunque el terminal y red permitan una mayor calidad, el usuario puede decidir un modo de menor calidad mientras no pasa nada de su interés).

Para la creación e intercambio de las descripciones, parte fundamental del sistema, se ha hecho uso del estándar XML (más en concreto MPEG-7), por lo que se incluyen diversos anexos en los que se puede consultar información detallada sobre XML y MPEG-7. Mediante XML se tiene un formato universal, capaz de ser utilizado en cualquier plataforma y terminal cliente, al mismo tiempo que permite la portabilidad de la información. Se utilizará XML como formato de intercambio de datos e información siempre que sea posible. En concreto se hará uso de los estándares de descripción multimedia MPEG, basados en XML.

Al mismo tiempo, se ha utilizado C++ como lenguaje para programar las aplicaciones. Con este lenguaje de programación podremos realizar rutinas que tengan una alta carga computacional de una manera más rápida que con otros lenguajes de programación (como por ejemplo Java).

Después de seleccionar las tecnologías que usaremos deberemos diseñar una aplicación de escritura de descriptores MPEG-7, dicha aplicación generará todos los datos a transmitir pero en ningún momento se preocupará por la manera de implementar dicha transmisión. Esa tarea de

desarrollo e implantación no entra dentro de los objetivos del PFC. Los módulos de dicha aplicación serán descritos con detalle en las posteriores secciones.

En el otro extremo de la comunicación deberemos diseñar una aplicación que lea dichos descriptores MPEG-7 y junto con los contenidos visuales (fondo y texturas[7] de los objetos) que se transmiten, visualizar una aproximación de la información original en el terminal receptor.

La aplicación diseñada recogerá la información necesaria proveniente de las cámaras o dispositivos de captación de imágenes. Dichas imágenes deberán tener las siguientes características:

- Resolución espacial variable
- 8bpp (escala de grises)
- Formato codificación = BMP de Microsoft

En el caso de que las secuencias de imágenes obtenidas fuesen en otro formato (JPEG en color por ejemplo), se deberían procesar para convertirlas a escala de grises en formato BMP. Para ello se debería añadir un módulo que hiciese la transcodificación de un formato a otro.

3.2 Diseño de metadatos

Los metadatos consisten en información que describen los datos, contenidos multimedia, recursos,…. Esta información posteriormente será utilizada por otras aplicaciones y además permiten acceder a lo que describe (en este caso contenidos multimedia) de una manera más eficiente. En nuestro sistema nos servirá para poder realizar la reconstrucción de la secuencia en la aplicación receptora.

Existen dos tipos de datos que manejamos en nuestro sistema: datos internos al programa y datos externos. Los datos internos son aquellas estructuras definidas para estimar el movimiento, el fondo de una escena, características de las imágenes,… Los datos externos son el producto final que obtenemos después de todo el proceso de nuestra aplicación: descripciones (en ficheros MPEG-7) e imágenes en formato BMP.

En este apartado se analizará cómo es el formato que se ha adoptado para el intercambio de datos entre los diferentes agentes del sistema (aplicación transmisora y receptora). Respecto a los

[7] El término textura se suele utilizar en MPEG para referirse a las imágenes que pueden asociarse o "pegarse" sobre objetos definidos por su forma (o malla si se trata de contenido 3D), separando la descripción más abstracta de forma de la imagen asociada a la misma.

datos internos se han definido estructuras en C++ para almacenar la información necesaria para nuestra aplicación. Respecto a los datos externos, es decir las imágenes que representarán al fondo y a los objetos, no hemos definido ningún nuevo formato, usaremos para ellas el formato BMP de Windows. Para más información sobre dicho formato, véase el anexo **"B CODIFICACION BMP"**.

A continuación se expondrá el diseño de los metadatos internos y externos, explicando los motivos de las decisiones tomadas.

3.2.1 Metadatos internos

Estos serán los datos internos que manejara nuestra aplicación. Tanto la aplicación que transmite la información como la que recibe la información y la reconstruye usarán el mismo tipo de metadatos.

Necesitamos crear estructuras para los siguientes datos:

- Imágenes: nuestra aplicación lee y escribe imágenes del disco duro de un PC. Es evidente que tendremos que diseñar una estructura que permita almacenar la información necesaria de una imagen.
- Movimiento: el objetivo de nuestra aplicación es analizar el movimiento de los objetos para luego después transmitir la información de movimiento. Para guardar la información del movimiento en cada cuadro, objetos moviéndose a través de una secuencia,…generaremos diversas estructuras para el manejo de estos datos.

A continuación se procede a explicar brevemente las estructuras usadas.

3.2.1.1 Imágenes

De las imágenes guardadas en disco, solamente nos interesarán sus dimensiones y el valor de los píxeles.

Para un manejo más sencillo de las imágenes, se ha diseñado una clase en C++. Se ha optado por diseñar una clase debido a las opciones de encapsulación que nos proporciona. Además incorporaremos varios métodos para poder trabajar con las imágenes (tales como leer, escribir,….).

La clase diseñada es la siguiente:

```
class ImagenGti
{
public:
// ATRIBUTOS PUBLICOS
        long idImagen;
        long timestamp;
        long sizeX;
        long sizeY;
        int bitsPixel;
        TIPO_PIXEL *pixels;

// METODOS PUBLICOS
        ImagenGti(long sizeX, long sizeY, int bitsPixel);
        virtual ~ImagenGti();
        void Inicializar(long sizeX, long sizeY, int bitsPixel);
        void Copiar(ImagenGti *origen);
        void Restar(ImagenGti *im);
        void SetPixels(TIPO_PIXEL *origen);
        void SetPixels(TIPO_PIXEL **origen);
        void SetPixels(TIPO_DCT **origen);
        int getBPP();
        int getSizeX();
        int getSizeY();
        bool CargarBmp(char *filename, long id);
        bool GuardarBmp(char *filename);
        void binarizar(TIPO_PIXEL umbral);
        void resize(ImagenGti *origen, int factor);
};
```

Figura 3-1: Definición de los metadatos internos asociados a las imágenes.

3.2.1.2 Movimiento

Respecto al movimiento tenemos varios aspectos que analizar:

- Movimiento de cada cuadro
- Movimiento entre cuadros consecutivos
- Identificación y seguimiento de objetos en una secuencia de imágenes de vídeo-seguridad

Para los metadatos asociados al movimiento en cada cuadro utilizaremos una estructura como la definida anteriormente, es decir, lo trataremos como si fuera una imagen. En dicha imagen guardaremos con el valor “0”(negro) todo aquello que pertenezca al fondo y con el valor “255” todo aquello que pertenezca al movimiento del cuadro actual. Otra imagen será usada para identificar los objetos, en este caso en vez de usar el valor “255” para los objetos, usaremos los

valores 1,2,3,4,… para identificar los distintos objetos del cuadro (con esta decisión identificaremos los distintos objetos de cada cuadro).

Para los metadatos asociados al movimiento entre cuadros consecutivos definiremos una estructura como la siguiente:

```
typedef struct{

        int cuadro_act;
        int num_objetos; // que podran ser cuadrados, polinomiales...etc
        int *asociacion;

}CUADRO_ASOCIACION;
```

Figura 3-2: Definición de los metadatos internos para la asociación

de movimiento entre cuadros consecutivos

En dicha estructura guardaremos el cuadro con el que se corresponden los datos almacenados, los objetos identificados y una matriz de asociación de objetos. Dicha matriz de asociación de objetos contendrá para cada objeto (que indexará dicha matriz) un número que indicará a que objeto del cuadro anterior se corresponde.

Para los metadatos asociados a los objetos de interés que son analizados durante la secuencia de imágenes de vídeo-seguridad, utilizaremos la siguiente estructura:

```
typedef struct{
        int id;
        char *path;
        int tamX;
        int tamY;
        int cuadro_start;
        int cuadro_finish;
        char *FreeTextAnnotation;
        char *CreationTime;
        char *obj_duration;
        char *obj_timePoint;
        char *VisualDescriptor;
        //coordenadas (x e y) de los cuatro vertices del rectangulo (4) a lo largo de los cuadros
        COORDENADAS_CARTESIANAS **coord;

}CARACTERISTICAS_OBJETO;
```

Figura 3-3: Definición de los metadatos internos para un objeto

Con esta estructura nos será más fácil la generación de los descriptores de objetos. En ella guardaremos el tamaño de los objetos, la duración y coordenadas del movimiento y un pequeño descriptor visual del objeto.

3.2.2 Metadatos externos

En este apartado nos centraremos en los descriptores[8], para ellos se han utilizado formatos MPEG-7 que están basados en XML (concretamente en XML Schema).

Para describir las imágenes que analizarán los módulos de transmisión, se usará el estándar MPEG-7, que define unos descriptores y esquemas de descripción que proporcionan una estructura estándar de acuerdo a unos tipos de datos definidos por MPEG-7 y XML Schema.

Las herramientas que proporciona MPEG-7 para describir contenidos multimedia son inmensas (ya que MPEG-7 proporciona herramientas genéricas para tipos de aplicaciones muy diversas) y algunas bastante complejas, y permiten describir el contenido de diferentes formas, proporcionando mucha información. Mucha de esta información es muy compleja y en una gran parte no será utilizada, ya que se sale del ámbito de esta aplicación. Por ello es necesario restringir estas herramientas MPEG-7 (descriptores y esquemas de descripción) a los que cumplen con la funcionalidad necesaria. MPEG-7 proporciona, al igual que para otros estándares de la familia, el concepto de perfil (y nivel), que es un subconjunto de todas las herramientas definidas en el estándar necesarias para ciertos tipos de aplicaciones. En cualquier caso, para nuestra aplicación el subconjunto de herramientas necesario no coincide con ninguno de los perfiles definidos (unos por exceso y otros por defecto).

A la hora de diseñar los metadatos correspondientes a los datos de interés en nuestra aplicación nos fijaremos en los tipos de datos de los que disponemos: imágenes de fondo e imágenes de objetos segmentados.

A estos dos tipos de datos les tendremos que añadir otro tipo de metadatos más general (versión, datos de creación, etc.) que también irá codificado en MPEG-7.

3.2.2.1 Datos generales

Adicionalmente a los datos que manejará nuestra aplicación (imágenes y objetos en movimiento) incluiremos información sobre distintos aspectos relevantes. Dicha información irá en

[8] Cuando nos referimos al término descriptores, nos estamos refiriendo a los metadatos asociados al contenido multimedia que es procesado por la aplicación.

las cabeceras de los ficheros que contendrán las descripciones de las imágenes de fondo o de los objetos.

Principalmente nos interesan las siguientes características:

- *Información sobre la versión del documento*

Esta parte es bastante común y suele encontrarse en casi cualquier documento XML. En el que encontraremos su *Namespace*[9], la versión del documento (con su fecha) y algún comentario adicional.

```
<Mpeg7 xmlns="urn:mpeg:mpeg7:schema:2001"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <DescriptionMetadata>
    <Version>1.0</Version>
    <LastUpdate>2006-09-20T12:20:25</LastUpdate>
    <Comment>
      <FreeTextAnnotation>
            write any text here
      </FreeTextAnnotation>
    </Comment>
    ……
</DescriptionMetadata>
</Mpeg7>
```

Figura 3-4: Definición de los metadatos externos comunes (descriptor XML)

- *Información sobre la creación del documento*

Aquí podremos encontrar todos los datos referentes al creador del documento, lugar de creación, instrumento de creación, derechos, …

```
<Creator>
      <Role href="creatorCS">
        <Name>Creator</Name>
      </Role>
      <Agent xsi:type="PersonType">
        <Name>
          <GivenName>Juan Carlos</GivenName>
          <FamilyName>San Miguel</FamilyName>
        </Name>
      </Agent>
</Creator>
<CreationLocation>
       <Region>es</Region>
       <AdministrativeUnit>Madrid</AdministrativeUnit>
</CreationLocation>
<CreationTime>2006-02-07T12:03</CreationTime>
```

Figura 3-5: Definición de los metadatos externos sobre creación del

[9] Para más información sobre *Namespaces* consultar el anexo sobre XML

documento (descriptor XML)

- *Información contextual de la imagen*

Con esta información pretendemos dar más cabida a una interpretación de lo que se está indexando mediante el documento. En esta estructura se explicarán detalles semánticos

```
<TextAnnotation>
          <FreeTextAnnotation>Juan Carlos, EPS UAM, ferbuary, for my PFC,
transmission at low rate, with C++</FreeTextAnnotation>
          <StructuredAnnotation>
            <Who>
              <Name>Juan Carlos</Name>
            </Who>
            <WhatAction>
              <Name>transmission at low rate</Name>
            </WhatAction>
            <Where>
              <Name>EPS UAM</Name>
            </Where>
            <When>
              <Name>ferbuary</Name>
            </When>
            <Why>
              <Name>for my PFC</Name>
            </Why>
            <How>
              <Name>with C++</Name>
            </How>
          </StructuredAnnotation>
        </TextAnnotation>
        <Semantic>
          <Label>
            <Name />
          </Label>
          <Graph />
        </Semantic>
```

Figura 3-6: Definición de los metadatos externos sobre información contextual de la imagen (descriptor XML)

3.2.2.2 Imágenes de fondo (background)

Para este tipo de datos solo nos interesa generar una pequeña descripción de la imagen.

Las características principales que queremos almacenar sobre las imágenes fijas de fondo son:

- Formato de la imagen
- Tamaño en píxeles
- Bpp
- Ruta de la imagen
- Comentarios y Anotación textual
- Descriptor Visual

Con todos estos atributos definiremos las imágenes de fondo que usa nuestra aplicación.

Respecto a este tipo de metadatos sólo quedaría por comentar un último detalle, el tipo de descriptor visual que usaremos para describir la imagen. Este atributo nos servirá para hacernos una idea del color[10] de la imagen que estamos codificando en MPEG-7.

Se corresponde con el tipo

```
<element name="VisualDescriptor" type="MPEG-7:VisualDType"/>
```

Para este atributo tenemos varias opciones (especificadas en la parte 3 de estándar MPEG-7: ISO/IEC 15938-3 Visual):

- *ColorLayoutType*
- *ScalableColorType*
- *EdgeHistogramType*
- *DominantColorType*

Hemos decidido usar *DominantColorType* ya que es una característica sencilla de obtener y para nuestra aplicación es suficiente con una sencilla descripción del color para identificar los objetos. Para una explicación más detallada consultar el anexo **"VISUAL DESCRIPTOR: DOMINANT COLOR".**

Todos estos atributos se detallaran en la siguiente figura:

[10] En la versión actual del software, al trabajar con imágenes de niveles de gris, realmente usamos el nivel de gris (lumninacia), pero el sistema está preparado para una sencilla migración a imágenes a color.

```
<Description xsi:type="ContentEntityType">
    <MultimediaContent xsi:type="ImageType">
      <Image>
        <MediaInformation>
          <MediaProfile master="true">
            <MediaFormat>
              <Content href="image" />
              <FileFormat href="urn:mpeg:MPEG7FileFormatCS:1">
                <Name>BMP</Name>
              </FileFormat>
              <VisualCoding>
                <Format href="urn:mpeg:MPEG7FileFormatCS:1" colorDomain="color">
                  <Name>BMP</Name>
                </Format>
                <Pixel bitsPer="8" />
                <Cuadro width="100" height="100" />
              </VisualCoding>
            </MediaFormat>
            <MediaInstance>
              <MediaLocator>
                <MediaUri>file:/C:/Escritorio/recorte.jpg</MediaUri>
              </MediaLocator>
            </MediaInstance>
          </MediaProfile>
        </MediaInformation>
        <CreationInformation>
                  <CreationTime>2006-02-07T12:03</CreationTime>
        </CreationInformation>
        <VisualDescriptor xsi:type="DominantColorType">
                  <SpatialCoherency>0</SpatialCoherency>

                  <Values>
                        <Percentage>15</Percentage>
                        <LuminanceColorValueIndex>55</LuminanceValueIndex>
                  </Values>
                  <Values>
                        <Percentage>3</Percentage>
                        <LuminanceValueIndex>100</LuminanceValueIndex>
                  </Values>
                  <Values>
                        <Percentage>2</Percentage>
                        <LuminanceValueIndex>150</LuminanceValueIndex>
                  </Values>
        </VisualDescriptor
      </Image>
 </MultimediaContent>
```

Figura 3-7: Definición de los metadatos externos para una imagen (descriptor XML)

3.2.2.3 Imágenes de objetos en movimiento

El tratamiento de los objetos en movimiento será similar al de las imágenes de fondo. Para cada objeto tendremos una imagen correspondiente, con la cual generaremos un descriptor similar? al anteriormente explicado. Además, nos interesará la trayectoria que describa el objeto. Esa será la característica principal que incluiremos en el descriptor XML.

Para los objetos en movimiento usaremos las herramientas de descripción de *MovingRegion DS*, descritas en el estándar MPEG-7 parte 5.

A la hora de generar los descriptores nos fijaremos en las siguientes características:

- Imagen del objeto durante la trayectoria (tratamiento igual que la imagen de fondo)
- Movimiento del objeto
    - Tiempo de inicio y fin del movimiento
    - Tiempo de actualización de objeto
    - Coordenadas de los vértices

La idea de este descriptor es representar mediante puntos (vértices) la región que representa al objeto. Después de identificar dichos vértices se procede a describir la trayectoria de cada uno (si el objeto es rígido es suficiente con describir la trayectoria de un vértice, pero en nuestro sistema no podemos asumir que el objeto es rígido debido a que la trayectoria puede implicar cambio de profundidad y el movimiento propio de los objetos puede hacer cambiar su silueta –e.g., persona caminando-).

Si el objeto es rectangular (caso en que representamos al objeto mediante la "bounding box" asociada al objeto en movimiento) entonces nos fijaremos en los cuatro vértices del rectángulo y describiremos como se van moviendo con el paso del tiempo, para así después poder representarlo en la aplicación receptora.

Si el objeto es poligonal la única diferencia es que tendremos un mayor número de puntos y entonces la descripción MPEG-7 correspondiente tendrá un tamaño mayor.

Un ejemplo de descripción será la siguiente (se ponen solamente dos vértices para simplificar el ejemplo):

```
<SpatioTemporalDecomposition>
                <MovingRegion id="Region1">
                        <TextAnnotation>
                          <FreeTextAnnotation>PFC</FreeTextAnnotation>
                       </TextAnnotation>
                        <SpatioTemporalLocator>
                                <FigureTrajectory type='rectangle'>
                                        <MediaTime>
                                                <MediaTimePoint>T00:00:15</MediaTimePoint>
                                                <MediaDuration>PT1M15S</MediaDuration>
                                        </MediaTime>
                                        <!-- Primer vertice
                                        -->
                                        <Vertex>
                                                <WholeInterval>
                                                        <MediaDuration>PT0S</MediaDuration>
                                                </WholeInterval>
                                                <!-- primera dimension (x)
                                                -->
                                                <InterpolationFunctions>
                                                        <KeyValue type="startPoint">0.0</KeyValue>
                                                        <KeyValue type="firstOrder">2.5</KeyValue>
                                                        <KeyValue type="firstOrder">5.0</KeyValue>
                                                        <KeyValue type="firstOrder">7.5</KeyValue>
                                                </InterpolationFunctions>
                                                <!-- segunda dimension (y)
                                                -->
                                                <InterpolationFunctions>
                                                        <KeyValue type="startPoint">0.0</KeyValue>
                                                        <KeyValue type="firstOrder">2.5</KeyValue>
                                                        <KeyValue type="firstOrder">5.0</KeyValue>
                                                        <KeyValue type="firstOrder">7.5</KeyValue>
                                                </InterpolationFunctions>
                                        </Vertex>
                                        <!-- Segundo vertice
                                        -->
                                        <Vertex>
                                                <WholeInterval>
                                                        <MediaDuration>PT0S</MediaDuration>
                                                </WholeInterval>
                                                <!-- primera dimension (x)
                                                -->
                                                <InterpolationFunctions>
                                                        <KeyValue type="startPoint">0.0</KeyValue>
                                                        <KeyValue type="firstOrder">2.5</KeyValue>
                                                        <KeyValue type="firstOrder">5.0</KeyValue>
                                                        <KeyValue type="firstOrder">7.5</KeyValue>
                                                </InterpolationFunctions>
                                                <!-- segunda dimension (y)
                                                -->
                                                <InterpolationFunctions>
                                                        <KeyValue type="startPoint">0.0</KeyValue>
                                                        <KeyValue type="firstOrder">2.5</KeyValue>
                                                        <KeyValue type="firstOrder">5.0</KeyValue>
                                                        <KeyValue type="firstOrder">7.5</KeyValue>
                                                </InterpolationFunctions>
                                        </Vertex>
                                </FigureTrajectory>
                        </SpatioTemporalLocator>
                       <VisualDescriptor xsi:type="DominantColorType">
                                <SpatialCoherency>0</SpatialCoherency>
                                <Values>
                                        <Percentage>15</Percentage>
                                        <ColorValueIndex>55</ColorValueIndex>
                                </Values>
                                ...
                    </VisualDescriptor>
                </MovingRegion>
            </SpatioTemporalDecomposition>
```

Figura 3-8: Definición de los metadatos externos para un objeto (descripción MPEG_7)

3.3 Arquitectura del sistema

3.3.1 Introducción

El sistema constará de los siguientes elementos:

- *Cámara de vídeo:* que capta las secuencias de imágenes
- *Aplicación transmisora:* que analizará la secuencia de imágenes, separando en fondo y los objetos en movimiento durante la secuencia.
- *Transmisor*: elemento del sistema que transmitirá los datos generados por la aplicación transmisora
- *Receptor:* elemento del sistema que recibirá todos los datos transmitidos.
- *Aplicación Receptora:* que a partir de los datos recibidos realizara una reconstrucción de la secuencia original
- *Presentación*: donde se podrán ver los resultados (visualizando un vídeo en MPEG2)

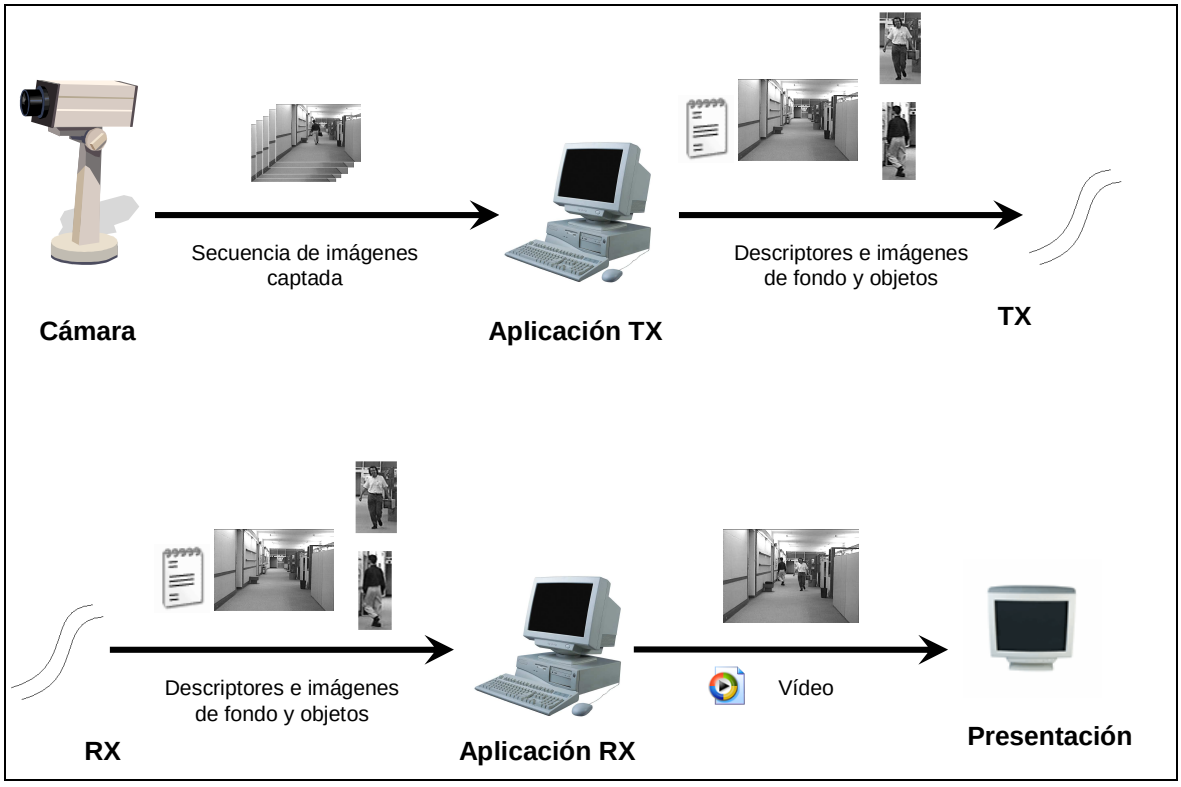


Figura 3-9: Arquitectura general del sistema

En los posteriores apartados se analizarán los módulos a desarrollar dentro del presente PFC: la aplicación transmisora y la aplicación receptora.

3.3.2 Aplicación transmisora

Esta aplicación es la encargada de generar toda la información a transmitir. Para ello partirá de la secuencia de imágenes procedentes de una cámara (caso de uso real) o fichero.

Dichas imágenes estarán en formato BMP (por lo que en caso de otro formato existirá un módulo de transcodificación), de tamaño variable y en escala de grises. El esquema de dicha aplicación se expone en la figura 3-10.

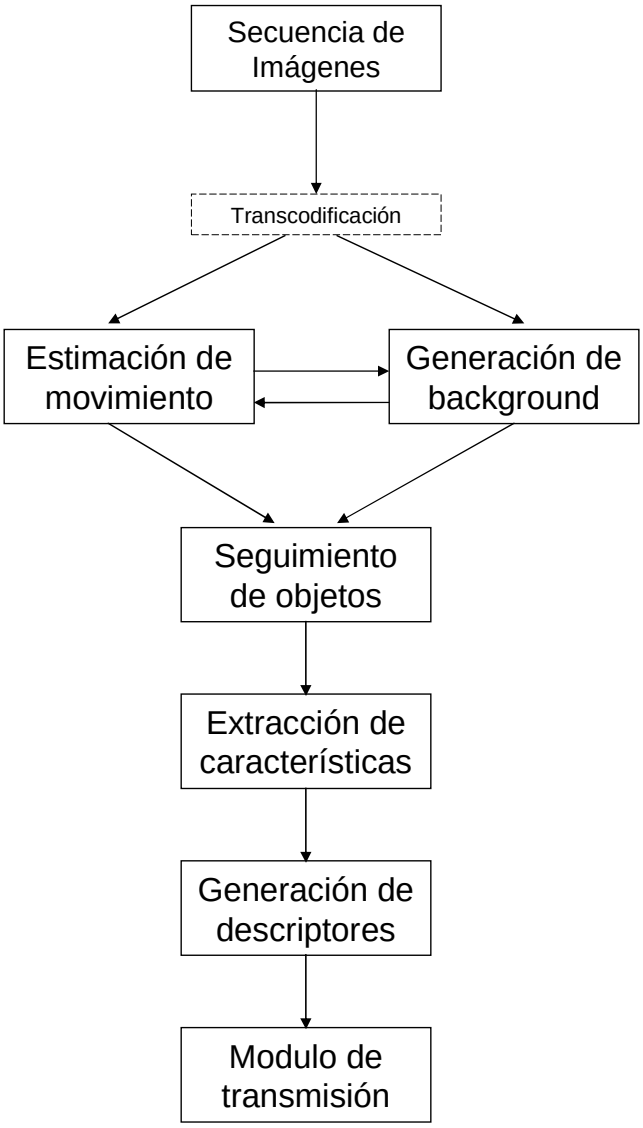


Figura 3-10: Aplicación Transmisora

A la hora de implementar dichos módulos hay que destacar que el objetivo inicial es que sean aplicaciones en "tiempo-real" y que su diseño no tiene que estar tan orientado a la máxima

exactitud de la detección de los objetos y el movimiento detallado, como en la rapidez con la que se ejecuten los algoritmos, sin que esto signifique que el error sea excesivamente grande. Esto está justificado por la aplicación a la que se orienta el sistema, que se centra en una transmisión a baja tasa, no a una segmentación perfecta.

A continuación se pasan a explicar los diferentes módulos de la aplicación y las decisiones de diseño.

3.3.2.1 Módulo de estimación de movimiento

Para poder generar los descriptores asociados al movimiento en dichas imágenes tendremos que detectar en primer lugar el movimiento en la secuencia de imágenes que llega a nuestro sistema.

La tarea fundamental del módulo será la de detectar lo objetos en movimiento dentro de una secuencia de imágenes. La decisión de si un objeto pertenece al movimiento o no deberá ser tomada en función a ciertos parámetros, como puedan ser tamaño del objeto, duración del movimiento,…para así poder evitar errores (falsos positivos, falsos negativos) en la identificación de objetos.

Además de la detección de los objetos, dicho módulo deberá también almacenar una máscara en la que se incluya toda la información de movimiento detectada. Dicha máscara será una imagen binaria (blanco y negro): se representará en color blanco aquellos objetos (puntos) en movimiento y en color negro el resto de la imagen. Esta máscara nos servirá como datos de entrada para los siguientes módulos (seguimiento y background).

El formato de la máscara (imagen que contiene los objetos) será el mismo que la imagen de entrada. Es decir, las imágenes de salida de este algoritmo serán imágenes en formato BMP y en escala de grises (8bpp), aunque se esté en algunos casos desperdiciando posibilidades de compresión (e.g., para la máscara binaria es suficiente con una imagen de 1 bpp).

El sensor de movimiento deberá tener una estimación del movimiento píxel a píxel, mientras que las imágenes de salida que identifican a los objetos identificados en la secuencia podrán contener píxeles que no son pertenecientes al movimiento. Este hecho se debe a que si se detecta un objeto y después lo queremos encuadrar para un posterior procesado (como objetos cuadrados), estaremos haciendo que los píxeles colindantes con el objeto pasen a formar parte del objeto.

Así pues distinguiremos dos tipos de objetos de salida:

- Objetos rectangulares:

Dichas imágenes se ajustan de manera rectangular a los objetos identificados por el módulo. Así pues las imágenes de salida contienen los objetos enteros (si no hay causas que hagan confundir el objeto con el fondo –e.g. color de objeto similar al fondo-) y parte del fondo de la imagen de la que partíamos.

Figura 3-13-11: Segmentación en objetos rectangulares

- Objetos "polinomiales":

  Las imágenes se ajustaran de manera polinomial a los objetos identificados, obteniendo un ajuste mucho más fino. Es decir, nuestra aprocimación a las imágenes objeto será muy parecida a los objetos identificados (siempre que el algoritmo de segmentación funcione correctamente).

Figura 3-12: Segmentación en objetos polinomiales

Con lo cual nuestro detector de movimiento deberá soportar ambos modos de funcionamiento, así el usuario podrá elegir aquel que se ajuste más a sus preferencias.

Por ultimo habría que determinar un parámetro más en el diseño de este módulo, la frecuencia con la que obtenemos la información del movimiento de la secuencia.

Dicho parámetro lo llamaremos *Tiempo de Actualización de Objeto* (TAO), y nos indicará cada cuantos segundos tendremos que extraer información de los objetos en movimiento de la secuencia. Este parámetro deberá ser configurable a gusto del usuario.

Para ver los detalles de los algoritmos desarrollados véase la sección 4.1.2.

3.3.2.2 Módulo de estimación de fondo

La necesidad de este módulo es debida a la reconstrucción que se va a producir en la aplicación receptora. Dicha aplicación receptora cogerá el fondo estimado por este módulo (y posteriormente transmitido) e irá dibujando encima los objetos que se van moviendo conforme a los descriptores de movimiento también transmitidos.

Así pues el objetivo de este módulo es generar una imagen de fondo donde se puedan visualizar los objetos que se van moviendo con el paso del tiempo.

Dicha generación de imagen de fondo se realizará cada cierto tiempo, dicho tiempo será parametrizable y se llamará *Tiempo de Actualización de Fondo* (TAF). Con este parámetro lo que queremos indicar es cada cuanto tiempo nuestro sistema extraerá información de la secuencia de imágenes y generará una imagen de fondo.

Para generar una imagen de fondo continua, dicho módulo necesita información proveniente de la detección del movimiento en la imagen de la cual sacaremos el fondo. A partir del movimiento y de la imagen en sí, se genera una nueva imagen con las mismas dimensiones que la imagen original en la que en vez de los objetos en movimiento detectados se ponen una serie de ponderaciones provenientes del análisis de secuencias de imágenes anteriores (para detalles del algoritmo ver sección 4.1.1).

3.3.2.3 Módulo de seguimiento de movimiento

La tarea principal de este módulo será el seguimiento ("*tracking*") de los objetos que son detectados en la secuencia de imágenes analizada.

La frecuencia que con la que se obtendrán imágenes con información de movimiento vendrá determinada por el parámetro *Tiempo de Actualización de Objeto* (TAO) y dichas imágenes serán obtenidas del módulo segmentador.

Para realizar la tarea de seguimiento y con el fin de distribuir la carga de proceso de dicho módulo, se ha dividido en distintas subtareas:

- Asociación de objetos entre cuadros consecutivos

La tarea será la de asociar los objetos que aparecen entre cuadros consecutivos (cuadros obtenidos cada *Tiempo de Actualización de Objeto*). Esta tarea es sencilla de implementar, simplemente necesitará dos cuadros y el número de objetos que hay en cada uno.

Su función será la de asignar a cada objeto, su objeto correspondiente en el anterior cuadro. En caso de no existir, esta asignación quedaría vacía.

Con este módulo se pretende realizar un enlace cuadro a cuadro del movimiento de un objeto (la trayectoria de dicho objeto será construida con esta información en el módulo posterior).

Para asociar el movimiento se definirá una estructura sencilla que contendrá:

  - Cuadro actual
  - Número de objetos detectados (cuadrados, polinomiales,….dependiendo del modo en el que trabajemos)
  - Una asociación de los objetos del cuadro actual con el anterior.

- Seguimiento de los objetos en la secuencia analizada

Esta tarea consistirá en identificar y seguir los objetos de interés en toda la secuencia de imágenes analizada.

Para ello partiremos de las asociaciones cuadro a cuadro realizadas anteriormente y de las restricciones que nos impondrá el usuario para detectar objetos de interés (tamaño, duración,…).

Los datos generados en dicho análisis se utilizarán posteriormente para generar los descriptores de movimiento.

3.3.2.4 Módulo de extracción de características

La tarea de este módulo es extraer las características necesarias para poder generar las descripciones MPEG-7 que posteriormente serán transmitidas.

El objetivo de este módulo es dar soporte al módulo explicado en el siguiente apartado, así que sus funciones estarán definidas por las características que queramos plasmar en las descripciones de las imágenes.

Para la escritura de los descriptores necesitamos tres tipos de características:

- Características generales
    - Nos interesarán datos como nombre, versión, formato del documento, última actualización, creador, lugar de creación, inicio de tiempo de creación del documento, nombre de la herramienta, derechos,…
- Características de las imágenes de fondo
    - Nos interesarán características como el nombre del fichero, formato del fondo, tamaño en píxeles de la imagen, comentarios, descriptores visuales, bits por píxel,…
- Características de los objetos
    - Para los objetos inicialmente nos interesarán las mismas características que para el fondo (puesto que ambos son imágenes). La única característica adicional será la información referente al movimiento y la forma del objeto.
    - Respecto al movimiento nos interesará la siguiente información:
        - Cuadro en el que se inició/terminó el movimiento
        - Coordenadas del movimiento
        - Imagen que representa a cada objeto
        - Duración e inicio del movimiento del objeto
        - Descriptores visuales del objeto

Así pues para la extracción de dichas características dicho módulo necesitará todas las estructuras procesadas en los módulos anteriores; tanto las imágenes de partida como las distintas estructuras de información que irán generando los distintos procesos que se ejecuten secuencialmente (cálculo de la imagen de fondo, estimación de movimiento, seguimiento de los objetos,…).

Para ver todas las estructuras de información que se generan en este módulo con más detalle, ver la sección 3.2.2

Adicionalmente cuando dicho módulo extraiga las características de los objetos que se detectan en la secuencia de imágenes, guardará también una imagen que represente a cada objeto durante todo el análisis de la secuencia hasta que se vuelva a repetir el análisis de otra secuencia de imágenes (que coincidirá con cada actualización de fondo (TAF –ver sección 3.2.2.2)).

3.3.2.5 Módulo de generación de descripciones

Este módulo se encarga de generar la descripción maestra y las diversas descripciones que se generarán en función del nivel de transmisión que queramos.

Las descripciones se guardarán en formato MPEG-7 en ficheros con extensión mp7. Estos ficheros serán escritos en disco para su posterior manipulación por el módulo de transmisión.

Con respecto a las descripciones, su contenido estará jerarquizado por niveles. Estos niveles (también llamados niveles de transmisión) estarán organizados en función a los requerimientos del terminal o del medio sobre el que se transmitirán los datos que genere la aplicación. Así los niveles serán los siguientes:

- **Nivel 1 (descripción maestra)**: descripción MPEG-7 completa, incluyendo todas las imágenes asociadas a cada objeto (cada *Tiempo de Actualización de Objeto*) y un fondo variable cada TAF (*Tiempo Actualización de Fondo*) segundos. Esta descripción se generará cada TAF y se guardará en un único fichero mp7 ("Description_1.mp7").
- **Nivel 2**: descripción MPEG-7 completa, una imagen de fondo y una imagen asociada a cada objeto cada TAF (*Tiempo Actualización de Fondo*) segundos.
- **Nivel 3**: descripción MPEG-7 completa y una imagen de fondo. Para representar los objetos nos basaremos en los descriptores visuales que encontraremos dentro de la descripción MPEG-7. Es decir representaremos a cada objeto con su luminancia dominante.

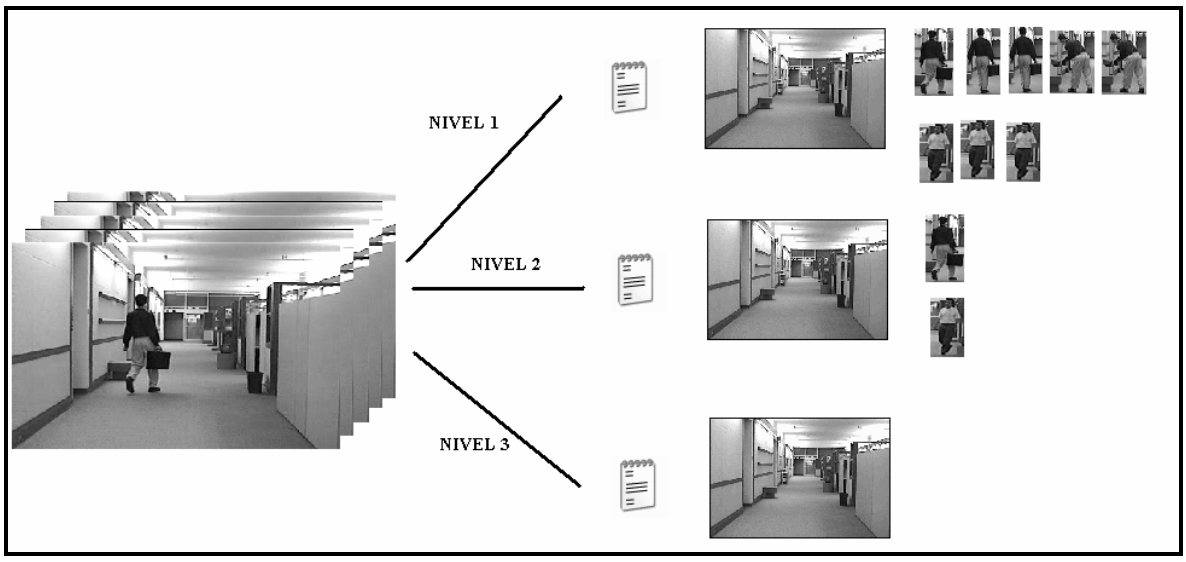


Figura 3-13: Niveles de generación de descriptores

Para la generación de las descripciones, este módulo partirá de los datos que extraemos del módulo anterior (Extracción de Características) y dará como salida la descripción maestra y las secundarias en función de si las hemos requerido o no.

3.3.2.6 Módulo de transmisión

El módulo de transmisión es el encargado de transmitir toda la información que se genere en el módulo anterior. Para ello utilizará diversas técnicas de transmisión en función del canal que utilice o de los requisitos del terminal transmisor/receptor.

La tarea de desarrollo e implantación no entra dentro de los objetivos del PFC. Con lo cual no entraremos en detalle sobre los requerimientos de diseño de dicho módulo de la aplicación. Simplemente nos bastará con que simule una transmisión de los datos para que un módulo receptor (en la aplicación receptora) sea capaz de reconstruirlos.

Los elementos que tendrán que ser transmitidos por dicho módulo serán los siguientes:

- Descripciones (una por nivel) MPEG-7 asociadas a la secuencia a transmitir (ficheros XML con extensión mp7).
- Imágenes de fondo en formato BMP
- Imágenes que contienen las texturas de los objetos en movimiento (también en formato BMP)

Todos estos datos se dispondrán en una carpeta a la que tendrán acceso el módulo de transmisión y el de recepción.

En el diseño de dicho módulo hay que destacar que el tiempo que tarda en procesar y enviar todos los datos ha de ser menor al que tarde nuestro sistema en producir nuevos datos (nuevos descriptores, actualizaciones de fondo, nuevos objetos,…) si queremos cumplir con el objetivo de tiempo real. En cualquier caso, al no entrar dentro de los objetivos del PFC el desarrollo de un sistema real de transmisión-repceción, este módulo se encargará de simular este paso y de evaluar el coste de transmisión (tamaño de ficheros, tasa binaria, …)

3.3.3 Aplicación Receptora

3.3.3.1 Introducción

Además del desarrollo de un módulo que genere los descriptores y las imágenes necesarias (objetos y fondo), se deberá desarrollar también otra aplicación para la simulación de la recepción y la visualización de los datos enviados.

Esta aplicación ha de ser sencilla y rápida pues lo único que queremos es mostrar (en caso de que el usuario así lo requiera) la imagen de fondo con objetos moviéndose superpuestos en la imagen de fondo. El número de imágenes enviadas y el grado de detalle del movimiento de los objetos vendrá determinado por el tipo de nivel de transmisión utilizado, que se indicará en el documento MPEG-7 que se recibirá junto a las imágenes. Es decir, la aplicación receptora primeramente lo que tendrá que hacer es analizar el documento mp7 que recibe y después mostrar una simulación de aquello que ha sido captado por la cámara y que nuestra aplicación transmisora ha procesado y enviado.

La salida de esta aplicación será un video en MPEG-2[11] que contendrá la secuencia de imágenes que se generarán cada tiempo de actualización de fondo.

Así pues el sistema de recepción queda de la siguiente manera:

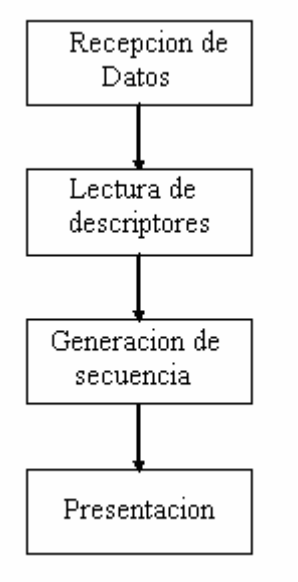


Figura 3-14: Esquema general de la aplicación receptora

3.3.3.2 Módulo de Recepción

[11] Podría haberse elegido cualquier otro formato.

Este módulo es el encargado de recibir los datos transmitidos y decodificarlos para su posterior procesado en los siguientes módulos.

La tarea de desarrollo e implantación no entra dentro de los objetivos del PFC. Con lo cual no entraremos en detalle sobre los requerimientos de diseño de dicho módulo de la aplicación

Su principal función será la de colocar en una carpeta todas las imágenes dejarlos en una carpeta en el mismo formato en el que estaban antes de ser transmitidos, es decir, imágenes en formato BMP y un fichero con extensión mp7.

La carpeta se denominará “rx” y en ella se dispondrá de tres subcarpetas (nivel 1, 2 y 3) en función de los niveles de transmisión recibidos. Dentro de cada subcarpeta se dispondrán los datos de igual manera que fueron generados. Es decir por cada secuencia analizada un descriptor de movimiento, una imagen de fondo e imágenes de objetos (si procede).

3.3.3.3 Módulo de Lectura de Descripciones

La tarea de este módulo es sencilla: deberá ir leyendo el fichero de texto (en formato MPEG-7) hasta obtener todos los datos necesarios para poder reconstruir la secuencia de imágenes original en función del nivel de transmisión recibido (1, 2 ó 3).

Los datos que nos interesarán para la reconstrucción de la secuencia serán:

- Referente al fondo
    - Ruta de la imagen
    - Dimensiones de la imagen
- Objetos
    - Imágenes de los objetos (una o varias en función del nivel de transmisión)
    - Dimensiones de los objetos
    - Inicio y fin del movimiento del objeto.
    - Luminancia dominante de las imagen de cada objeto
    - Movimiento de cada objeto en la secuencia analizada

El resto de información contenida en el fichero .mp7 deberá ser obviada para esta tarea.

A la hora de decidir qué nivel de reconstrucción se usará, este no se detectará automáticamente sino que deberá ser indicado por el usuario en la aplicación receptora, pudiéndose indicar uno o varios niveles de reconstrucción.

3.3.3.4 Módulo de generacion de secuencia

La tarea principal de este módulo será la de generar una reconstrucción de la secuencia de imágenes original en la aplicación Transmisora.

Para realizar dicha tarea, se generará una secuencia de imágenes con la imagen de fondo obtenida de dicha secuencia y dependiendo de los descriptores de movimiento, superpondrán distintos objetos en las imágenes.

En la superposición de las imágenes, lo que se pretende es que sobre la imagen de fondo se muestren las formas e imágenes de los objetos que han sido detectados en movimiento. Así al ver la secuencia iremos viendo la trayectoria que va describiendo cada objeto.

Evidentemente esta reconstrucción no será idéntica a la secuencia de imágenes inicial de la que se partía, lo único que se pretende es crear una simulación del movimiento que realiza el objeto identificado a través de la secuencia analizada.

Así pues dependiendo del nivel en el que nos encontremos, la reconstrucción se realizará de manera diferente:

- En el nivel 1, se partirá de una imagen fondo y una secuencia de imágenes para cada objeto. Dichas imágenes se superpondrán en la imagen de fondo en base a las coordenadas obtenidas de los descriptores de movimiento.

- En el nivel 2, se partirá de la imagen de fondo y una imagen por cada objeto identificado en la secuencia. Dicha imagen se irá superponiendo en los distintos cuadros según las coordenadas obtenidas de los descriptores de movimiento.

- En el nivel 3, se partirá de la imagen de fondo y las descripciones de movimiento. En lugar de colocar la imagen de cada objeto, podremos en su lugar un rectángulo con el color dominante de cada objeto.

En la siguiente figura se muestra un ejemplo de los resultados que se deberían obtener en la fase de reconstrucción:

Imagen original (cuadro 100) Imagen reconstruida nivel 1 (cuadro 100)

Imagen reconstruida nivel 2 (cuadro 100) Imagen reconstruida nivel 3 (cuadro 100)

Figura 3-15: Niveles de reconstrucción en el receptor

En dicha figura se puede observar que con los niveles de reconstrucción 2 y 3 se tiene una perdida de calidad (subjetivamente) bastante grande. En cambio en el nivel 1 (donde para cada cuadro de análisis de movimiento tenemos una imagen de objeto) es bastante complicado encontrar diferencias entre la secuencia original y la reconstruida.

3.3.3.5 Módulo de presentación

La tarea de este módulo es sencilla: generar un vídeo a partir de una secuencia de imágenes.

La secuencia de imágenes provendrá del módulo anterior y dicho módulo tendrá que dar como salido un video en MPEG2. El formato de las imágenes de entrada será BMP.

4 Desarrollo

Este capítulo se centra en cómo se ha llevado a cabo la implementación de los diseños de los diversos subsistemas que se han detallado en el capítulo anterior.

En primer lugar se tratará el desarrollo de los módulos de la aplicación de transmisión y posteriormente se procederá al desarrollo de la aplicación de recepción.

4.1 Desarrollo de la aplicación de transmisión

Al igual que en el capítulo de diseño se hizo un desarrollo modular de la aplicación de transmisión, a la hora del desarrollo de dichos módulos también se hará de manera modular. En el apartado 5 se analizará todo el sistema en conjunto y su comportamiento.

4.1.1 Generación de fondo

La generación del background (fondo) de una secuencia de imágenes es una tarea que influirá en el resto de módulos y en el procesado de la información.

Para la generación de fondo existen varias opciones (ver sección 2.1.2). Se han implementado dos de ellas, que no son las mas eficientes en cuanto a exactitud de generación del fondo, pero si en cuanto al consumo de recursos (tanto como en espacio que ocupa en memoria como en rapidez de ejecución).

A continuación se explican los dos métodos desarrollados, unas breves pruebas con cada uno de ellos y los problemas encontrados en cada algoritmo.

4.1.1.1 Algoritmo 1

Este algoritmo partirá de dos secuencias de imágenes proporcionadas por el detector de movimiento.

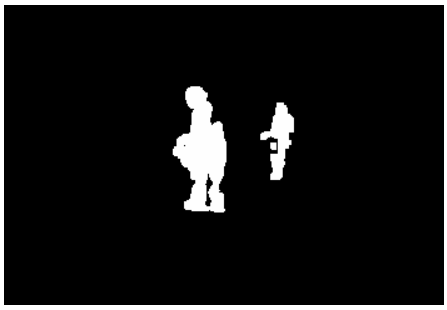


Imagen de la secuencia I(t) Imagen de la secuencia M(t)

Figura 4-1: Imágenes proporcionadas por el segmentador

La secuencia I(t) será la secuencia de imágenes captada por la cámara de video. Su formato deberá ser BMP y en escala de gris (8bpp).

La secuencia M(t) serán las máscaras binarias asociadas a la secuencia de imágenes. En dichas mascaras se representará cada píxel con dos valores: "255" (blanco) si el píxel se detecto como perteneciente a un objeto en movimiento o "0" (negro) si el píxel pertenece al fondo. Dichas máscaras provendrán de un procesado previo, que analiza la secuencia de imágenes y detecta los objetos móviles que encuentre en ella.

La duración de las secuencias es un parámetro que podremos establecer al inicio del cálculo de los fondos necesarios, siendo independiente la duración de las secuencias de la frecuencia con la que obtendremos el fondo.

Ambas secuencias tendrán de una dimensión fija, que será establecida como parámetro principal del programa de transmisión.

Inicialmente puede parecer que algo es incoherente que para generar el fondo usamos el propio fondo: la idea no es del todo incorrecta. Con las máscaras lo que tenemos es un fondo del momento actual t, y la salida de este algoritmo lo que persigue es generar un fondo general de la escena durante el tiempo que dure la secuencia de imágenes que lee el programa. Dicho fondo que se genera será una imagen completa (sin huecos correspondientes a objetos móviles) y nos servirá de referencia para "montar" objetos sobre ella en la aplicación transmisora.

Así pues la salida de este algoritmo será una imagen que represente el fondo para una secuencia de imágenes I(t) y M(t) dadas. Dicha imagen tendrá las mismas dimensiones que las imágenes de entrada y será en formato BMP y en escala de grises (8bpp).

El algoritmo se basa en la siguiente formula para generar la nueva imagen de background:

$$f(i,j) = \frac{\sum_{t=T-n}^{T} I(i,j,t) * M(i,j,t)}{\sum_{t=T-n}^{T} M(i,j,t)}$$

donde *I(i,j,t)* son los píxeles de la posición (i,j) del cuadro correspondiente al tiempo *t* y *M(i,j,t)* serán los píxeles correspondientes a la máscara de fondo (valor "0" si el píxel pertenece al

movimiento y “1” si pertenece al fondo). El tiempo *t* nos indicara con que imagen estamos trabajando dentro de la secuencia original (*t=0,1,2,3,…*).

Tal y como están definidos los datos en la fórmula, la secuencia de entrada M(t) (con valores “255” para indicar movimiento y “0” para indicar no movimiento) tendrá que ser procesada y cambiados sus valores (de “255” a “0” y de “0” a “1”) para su correcta aplicación al algoritmo.

Como se observa, para la generación de una imagen de fondo usaremos ‘n’ imágenes anteriores. Este parámetro deberá ser configurable por el usuario, al igual que el tamaño de las imágenes.

La idea de la fórmula es que para generar el fondo solamente se usen los píxeles pertenecientes a los fondos de las sucesivas imágenes de la secuencia. Luego después se le aplica una ponderación $\left(\sum_{t=T-n}^{T} M(i,j,t) \right)$ a dicha medida obtenida anteriormente, que es el número de veces que ha aparecido el píxel en el para no saturar el valor máximo que puede tener el píxel (255).

El algoritmo será el siguiente:

- **Para cada imagen de la secuencia (t=1,…..,T)**
    - **Leer imagen I(t) y M(t) de la secuencia**
    - **Modificar mascara M(t)**
    - **Si procede generar fondo en la iteración actual**
        - **Aplicar fórmula**
        - **Guardar en disco fondo**

Adicionalmente al desarrollo del algoritmo se dispuso un parámetro para indicar cada cuanto tiempo (el parámetro de medirá en cuadros y estos serán equivalentes a *Tiempo de Actualización de Fondo* * *cuadros_por_segundo*) se guardaba en disco el fondo generado con las secuencias anteriores.

El algoritmo se ha programado en un archivo ejecutable (“fondo.exe”) al que se le tiene que pasar los parámetros por línea de comandos. El formato de entrada será el siguiente:

```
fondo <sizeX> <sizeY> <NumImagenes> <ImagenesSalidaFondo> <ImagenesCalculoFondo>
```

- *sizeX* → tamaño horizontal de las imágenes de entrada (I(t) y M(t))
- *sizeY* → tamaño en vertical de las imágenes de entrada (I(t) y M(t))

- *NumImagenes* → número de imágenes de las que constan las secuencias
- *ImágenesSalidaFondo* → cada cuantas imágenes leídas de la secuencia queremos que se genere un fondo
- *ImagenesCalculoFondo* → cuantas imágenes anteriores queremos usar en la fórmula del calculo del fondo (parámetro "n" anteriormente explicado)

La salida de la aplicación, las imágenes que contienen el fondo, se guardarán en la misma carpeta donde están los datos originales.

4.1.1.2 Algoritmo 2

Para la realización de este algoritmo nos hemos basado en el método *Running Average* expuesto en el apartado *"2.1.4 Generación de fondo en secuencias de imágenes***"**

Este método se basa en ir actualizando el fondo a medida que vamos leyendo imágenes de la secuencia almacenada en el disco.

Como datos de entrada sólo necesita las imágenes de la secuencia y no las máscaras (de movimiento) de las imágenes. Este hecho supondrá una ventaja de rapidez de ejecución sobre el algoritmo 1.

El método *Running Average* sigue la siguiente formula:

$$B_{i+1} = \alpha * F_i + (1-\alpha) * B_i$$

donde B_{i+1} es el background nuevo que queremos calcular, B_i es el anterior background calculado y F_i es la imagen actual que estamos procesando.

El algoritmo se ha programado en un archivo ejecutable ("fondo2.exe") al que se le tiene que pasar los parámetros por línea de comandos. El formato de entrada será el siguiente:

```
fondo2 <sizeX> <sizeY> <NumImagenes> <ImagenesSalidaFondo> <alfa>
```

- *sizeX* → tamaño horizontal de las imágenes de entrada (I(t) y M(t))
- *sizeY* → tamaño en vertical de las imágenes de entrada (I(t) y M(t))
- *NumImagenes* → número de imágenes de las que constan las secuencias
- *ImágenesSalidaFondo* → cada cuantas imágenes leídas de la secuencia queremos que se genere un fondo
- *alfa* → parámetro del método *Running Average*

La salida de la aplicación, las imágenes que contienen el fondo, se guardarán en la misma carpeta donde están los datos originales.

4.1.1.3 Pruebas y resultados

Para la realización de pruebas, hemos escogido la secuencia de imágenes Hall Monitor, que consta de 300 imágenes de dimensiones 352x240, en formato BMP y en escala de grises (8bpp).

Ambas aplicaciones han sido programadas para coger los datos de una misma carpeta del directorio raíz (llamada "prueba") y volcaran sus datos en una carpeta llamada "fondo". El formato de las imágenes de entrada será el siguiente:

- Para las imágenes de la secuencia → "imageX.bmp" donde X indica el número de imagen (1…N)
- Para las máscaras de la secuencia → "maskX.bmp" donde X indica el número de mascara asociada a la imagen correspondiente (1…N)

Procederemos a probar ambos algoritmos con dichas secuencias y observaremos cuales son los resultados y cual sería el mejor para ejecutar en nuestro sistema.

A la hora de probar los algoritmos vamos a establecer como parámetros de calidad el tiempo de ejecución, la calidad subjetiva de la imagen de fondo obtenida y el espacio que ocupa en memoria.

Las pruebas que realizaremos estarán orientadas a obtener unos niveles de calidad de las imágenes de fondo de salida parecidas con los dos algoritmos, entonces compararemos el tiempo de ejecución.

Con el concepto de calidad de imagen obtenida se quiere referir a cuanto de bueno es el fondo que obtenemos y que los objetos en movimiento que aparecen en la secuencia no se encuentren representados en dicho fondo.

Como ultimo detalle sobre las pruebas es que estas se realizaron sobre el siguiente equipo: Pentium 4 3.20GHz, con 512MB de memoria RAM y sistema operativo Windows XP.

4.1.1.3.1 Algoritmo 1

Primeramente probamos con la secuencia Hall Monitor, leyendo la secuencia entera (300) y actualizando el fondo para cada 100 imágenes leídas. En el cálculo del fondo usaremos las 50

imágenes posteriores . Para ello ejecutaremos el programa con la siguiente instrucción por línea de comandos:

fondo 352 240 300 100 50

La duración de la ejecución fue de 1.125 segundos y los resultados fueron:

Figura 4-2: Imágenes de fondo obtenidas con el algoritmo 1 de la secuencia Hall Monitor (actualización cada 100 cuadros) y con 50 imágenes para el calculo de la nueva imagen de background

Donde podemos ver un problema que en principio no era perceptible: si usamos pocas imágenes para la actualización del nuevo fondo, es posible que los objetos en movimiento detectados hagan que no tengamos información de fondo sobre determinados píxeles. Este efecto se percibe en las zonas que aparecen en color negro. Para solucionar este problema la mejor opción es calcular la actualización de fondo con un mayor número de imágenes (con el consecuente gasto en memoria y en proceso).

Para ver que número de imágenes sería el idóneo para obtener una imagen de fondo de buena calidad probaremos con los siguientes valores: 100, 150, 200 y 250 imágenes para calcular el nuevo fondo. Para que el algoritmo sea más rápido, solamente sacaremos dos imágenes de fondo para toda la secuencia entera. El resultado de las ejecuciones fue:

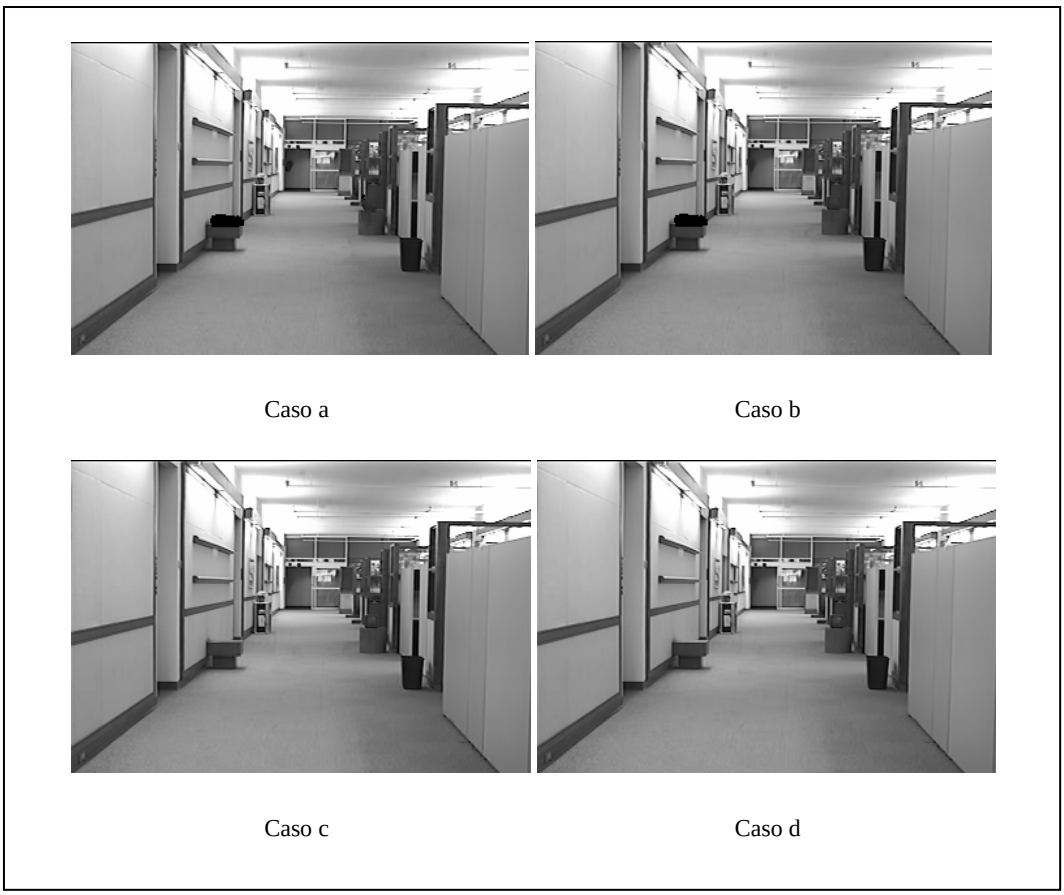


Figura 4-3: Imágenes de fondo obtenidas con el algoritmo 1 de la secuencia Hall Monitor (actualización cada 300 cuadros) y con a) 100, b) 150, c) 200 y d) 250 imágenes para el calculo de la nueva imagen de background. Los tiempos de ejecución en cada caso fueron de a) 1.21 sg , b) 1.45 sg, c) 1.65 sg y d) 1.89 sg

Como se observa en la figura 4-3, a partir de 200 imágenes conseguimos generar un fondo limpio. Aunque ya con 100 imágenes podríamos considerarlo limpio, ya que el objeto que aparece es debido al espurio que genera el módulo segmentador al detectar un objeto que se ha parado.

Para poder comparar resultados con el algoritmo 2, vamos a probar a lanzar la rutina para obtener una imagen de fondo (de cierta calidad) por cada imagen leída de la secuencia. Para ello ejecutamos la siguiente instrucción por línea de comandos:

fondo 352 240 300 1 200

El algoritmo tardó 174.84 segundos en ejecutarse y se obtuvieron imágenes de fondo de una calidad subjetiva media. En la siguiente figura se muestran algunos resultados:

Figura 4-4: Imágenes de fondo obtenidas con el algoritmo 1 de la secuencia Hall Monitor (actualización cada cuadro) y con 200 imágenes para el calculo de la nueva imagen de background. Las imágenes se corresponden con los cuadros a) 50, b) 100, c) 200 y d) 250

4.1.1.3.2 Algoritmo 2

Para el algoritmo 2 también probamos primeramente con la secuencia Hall Monitor: queremos obtener una secuencia de 3 imágenes de fondo con las 300 imágenes de la secuencia. Además buscamos un nivel de calidad similar al obtenido en el apartado anterior.

Para la primera prueba, ejecutamos la siguiente instrucción en la carpeta donde se encuentra la aplicación:

fondo2 352 240 300 100 0.05

Figura 4-5: Imágenes de fondo obtenidas con el algoritmo 2 de la secuencia Hall Monitor (actualización cada 100 cuadros) y con el parámetro alfa igual a 0.05

El tiempo de ejecución de dicho algoritmo fue de 0.35 segundos. La calidad obtenida de la imagen de fondo obtenida es bastante buena.

Para luego después poder comparar los resultados obtenidos con el algoritmo anterior, vamos a probar este algoritmo generando una imagen de fondo por cada cuadro que recibe. La instrucción que recibe seria la siguiente:

fondo2 352 240 300 1 0.05

En la siguiente figura vamos a mostrar los fondos generados cada 50 imágenes leídas:

Fondo correspondiente al instante 1

Fondo correspondiente al instante 50

Fondo correspondiente al instante 100

Fondo correspondiente al instante 150

Fondo correspondiente al instante 200

Fondo correspondiente al instante 250

Fondo correspondiente al instante 300

Figura 4-6: Imágenes de fondo obtenidas con el algoritmo 2 de la secuencia Hall Monitor (actualización cada frame) y con el parámetro alfa igual a 0.05

La ejecución del algoritmo duró 1.797 segundos y aunque el tiempo de ejecución es bastante rápido, se puede observar que las imágenes de fondo obtenidas tienen una calidad bastante baja (pues incluyen objetos móviles en el fondo).

4.1.1.3.3 Resultados y conclusiones

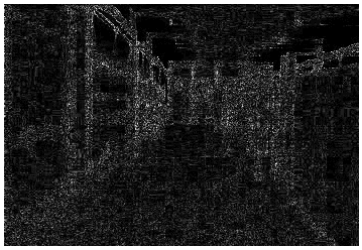
Como prueba de comparación de los fondos generados por ambos algoritmos, generaremos un mismo fondo con ambos algoritmos y realizaremos la diferencia con el original (que supondremos obtenido como primera imagen de la secuencia).

Como se puede observar en la figura 4.6, para el instante 10 la imagen de fondo generada con el algoritmo 1 es menos ruidosa que la generada con el algoritmo 2. Se aprecia una diferencia en toda la imagen (mayor en algoritmo 2 que en 1) cuando solo debería presentar diferencias en determinadas zonas donde ha cambiado la iluminación.

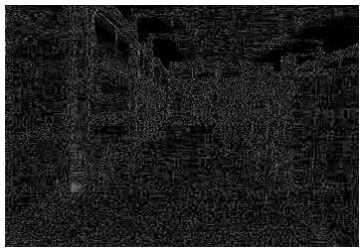
En cambio si avanzamos en el tiempo y observamos el fondo en el instante 100, observaremos que el fondo generado con el algoritmo 2 ha cambiado mucho más rápido que el algoritmo 1. Se ha incluido muy rápido un objeto que todavía esta moviéndose en la escena y que no debería ser incluido hasta que dicho objeto fuese identificado como objeto estático. Debido a este hecho, obtenemos una imagen de error mucho mayor que con el algoritmo 1.

Tal y como se observa en la figura, transcurrido un tiempo la calidad de las imágenes de fondo generadas por el algoritmo 1 es mayor que las del algoritmo 2.

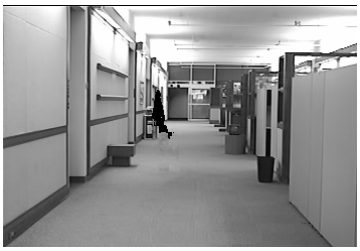
Fondo Original

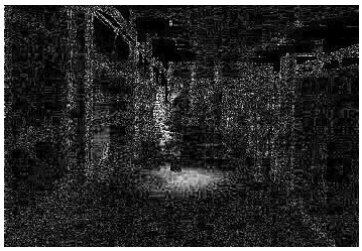


Fondo generado (t = 10) con Algoritmo 1 Diferencia con el fondo original

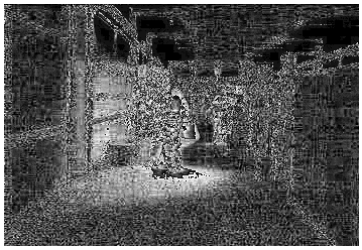


Fondo generado (t = 10) con Algoritmo 2 Diferencia con el fondo original





Fondo generado (t = 100) con Algoritmo 1 Diferencia con el fondo original



Fondo generado (t = 100) con Algoritmo 2 Diferencia con el fondo original

NOTA: los resultados de la imagen diferencia obtenidos han sido multiplicadas por 10 para así poder observar visualmente las diferencias con el cuadro original

Figura 4-7: Comparación de imágenes de fondo diferencia obtenidas con algoritmo 1 y 2

Respecto al espacio que ocupan en memoria, es evidente que el algoritmo 1 ocupará más espacio que el 2. El algoritmo 1 necesitará almacenar imágenes correspondientes a las secuencias *I(t)*, *M(t)* y al fondo. Mientras que el algoritmo 2 solo necesitará guardar imágenes correspondientes a la secuencia *I(t)* y las imágenes de fondo generadas.

Respecto al tiempo de ejecución claramente se observa que el algoritmo 2 es más rápido que el algoritmo 1, estos son los resultados obtenidos:

	Tiempo Algoritmo 1	**Tiempo Algoritmo 2**	**Calidad Algoritmo 1**	**Calidad Algoritmo 2**
Prueba 1: Obtener 3 imágenes de fondo de buena calidad de la secuencia completa (300 imágenes)	1.125 sg	0.35 sg	Si	Si
Prueba 2: Obtener una imagen de fondo por cada imagen leída	174.84 sg	1.797 sg	Si	Regular

Tabla 4-1. Comparación de resultados (algoritmos 1 y 2) para la generación y actualización de background

Respecto a la calidad de las imágenes obtenidas, el algoritmo 1 responde mejor a la presencia de objetos móviles durante largos periodos de tiempo. En cambio el algoritmo 2 responde muy rápido ante dichos objetos y los incluye rápidamente en la secuencia de fondo. Una posible solución para este problema seria la de reducir el factor alfa del algoritmo 2, inicialmente esta modificación funcionaría pero nos haría perder la actualización frente a los cambios de iluminación que aporta este algoritmo.

Todo esto nos sugiere realizar una mezcla de ambos algoritmos, combinar la eficiencia del algoritmo 1 con la rapidez del algoritmo 2. Esta es la solución utilizada para el sistema completo. Dicha solución se explica en el siguiente apartado.

4.1.1.4 Solucion utilizada

La solución que usaremos será la siguiente:

Utilizar el método *Running Average* pero no con todos los píxeles y así evitaremos incluir los objetos detectados por el segmentador. Para saber si el píxel pertenece al objeto en movimiento o no también necesitaremos la máscara de movimiento proporcionada por el módulo segmentador.

El algoritmo que usaremos será una mezcla del algoritmo 1 (usaremos su efectividad en el calculo de la imagen de fondo) y del algoritmo 2 (usaremos su rapidez de ejecución).

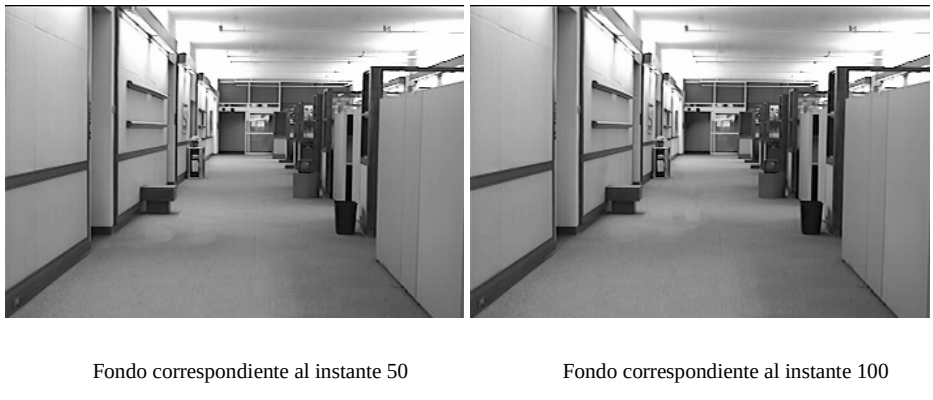
El algoritmo será el siguiente:

- **Para cada imagen de la secuencia**
  - **Leer imagen**
  - **Leer mascara generada en la iteración anterior**
    - **Generamos píxel a píxel el nuevo fondo**
      - **Si el píxel no perteneció a ningún objeto en la iteración anterior y la diferencia entre el cuadro actual y el anterior es pequeña (debida a cambios de iluminación por ejemplo)**[12]
        - **Aplicamos el método Running Average al píxel**
      - **En caso contrario procedemos de la siguiente manera**
        - **Contamos las veces que el píxel ha pertenecido al movimiento**
        - **Si ha pertenecido más de un determinado número de veces, consideramos que el objeto ha pasado al fondo**
        - **En caso contrario copiamos el píxel del fondo anterior**
  - **Guardar la imagen de fondo calculada si procede**

Respecto a las pruebas de test, no son comparables a los algoritmos anteriores. El problema es que para la generación de dicho fondo, necesitamos una estimación dinámica de las máscaras de movimiento y esta tarea consume un mayor tiempo de ejecución. Dicha generación de fondo vendrá integrada con el módulo segmentador y ambos módulos intercambiarán datos para realizar sus respectivas tareas.

Los resultados obtenidos fueron los siguientes:

[12] El hecho de observar la diferencia entre los píxeles del cuadro actual y el del anterior es para detectar un posible movimiento y así dejar al módulo segmentador que determine si finalmente ese cambio brusco se debió a la aparición de movimiento.



Fondo correspondiente al instante 50

Fondo correspondiente al instante 100

Fondo correspondiente al instante 150

Fondo correspondiente al instante 200

Fondo correspondiente al instante 250

Fondo correspondiente al instante 300

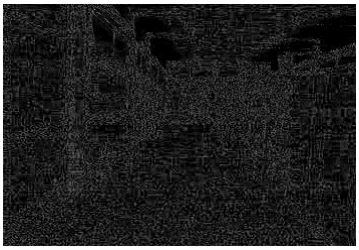
Figura 4-8: Imágenes de fondo obtenidas con el nuevo algoritmo de la secuencia Hall Monitor (actualización cada cuadro)

En la figura podemos observar que la calidad de las imágenes de fondo es bastante buena y el tiempo de ejecución es cualitativamente menor que el algoritmo 1 pero mayor que el del algoritmo 2 debido al mayor procesado que se hace de la imagen.

Para poder realizar una comparación con los algoritmos anteriores, vamos a calcular la imagen error en el instante t = 100 (que a una frecuencia de cuadro de 1 fps se correspondería con 100 cuadros leídos) y este fue el resultado:

Fondo Original

Fondo generado (t = 10)



Diferencia con el fondo original

Fondo generado (t = 100)

Diferencia con el fondo original

NOTA: los resultados obtenidos han sido multiplicadas por la constante 10 para así poder observar visualmente las diferencias con el cuadro original

Figura 4-9: Imágenes de error obtenidas con el nuevo algoritmo de la secuencia Hall Monitor (actualización cada cuadro)

Donde podemos observar que el error obtenido en la imagen diferencia es bastante menor que con los dos algoritmos anteriores.

4.1.2 Segmentación de objetos móviles

4.1.2.1 Algoritmo

En este apartado se mostrará el proceso seguido en el desarrollo del módulo de segmentación de imágenes. La segmentación que realizaremos será por detección de movimiento y con ella pretendemos detectar los objetos en movimiento que haya en la secuencia de imágenes que pretendemos analizar. Realmente este módulo no sólo se dedicará a la segmentación de objetos móviles sino que también realizará las tareas de actualización de background (fondo).

Para el desarrollo del detector de movimiento se ha seguido parte del algoritmo mostrado en [5] (ver sección 2.1.2). La decisión que tomaremos píxel a píxel se basará en la aplicación de la siguiente fórmula:

$$P\{g(i,j) \geq \tau(i,j)|H_0\} = \frac{\Gamma\left(\frac{q}{2}, \frac{g(i,j)^2}{2\sigma^2}\right)}{\Gamma\left(\frac{q}{2}\right)}.$$

y después comprobar si el valor obtenido es menor que un determinado umbral. En tal caso marcaremos dicho píxel como perteneciente a un objeto en movimiento, en caso contrario marcaremos dicho píxel como perteneciente al fondo (background).

La decisión se toma umbralizando el cuadro diferencia entre el cuadro actual y el cuadro que representa el fondo.

La imagen que representa el fondo se genera dinámicamente basándose en la información temporal mediante el algoritmo expuesto en la sección 4.1.1.4

El umbral nos servirá para descartar el efecto del ruido en la cámara después de la cuadro diferencia. Este valor se ha seleccionado empíricamente mediante pruebas (normalmente su valor oscilará entre 10^{-2} y 10^{-6}).

Para poder aplicar la fórmula tenemos que tener en cuenta los siguientes aspectos:

4.1.2.2 Funciones gamma ($\Gamma(a)$ y $\Gamma(a,x)$)

A la hora de aplicar la fórmula, nos surge la necesidad de aplicar la función gamma y la siguiente función gamma incompleta

$$\Gamma(a,x) = \int_x^\infty t^{a-1}\, e^{-t}\, dt.$$

Para una explicación mas detallada, consultar el siguiente anexo **"FUNCIONES TRASCENDENTALES".**

La implementación de dichas funciones es una tarea bastante complicada. En[34], se sugiere una implementación basada en aproximaciones de dichas funciones mediante series de potencias.

Dichas funciones son:

- *double gamma(double xx)* → cálculo de la función gamma
- *double gammq(double a, double x)* → cálculo de la función Q(a,x), con dicha función podemos calcular la función gamma incompleta que nos interesa

$$r(a, x) = r(a) * Q(a, x)$$

- *double gammp(double a, double x)* → cálculo de la función incompleta P(a,x)

Todas estas funciones se programarán en un fichero en lenguaje C, dicho fichero se llamará *"SpecialFunctions.c"*. En el fichero se detallan las instrucciones para poder obtener cualquiera de dichas funciones trascendentales.

4.1.2.3 Cálculo de la varianza del ruido

Otro de los parámetros que aparecen en la fórmula y que condicionará las decisiones que tome nuestro detector de movimiento es la varianza del ruido.

La varianza se define de la siguiente manera

$$Varianza = 1/N * \sum_i (x_i - x_m)^2$$

Para estimar la varianza del ruido nos vamos a centrar en el ruido introducido por la cámara. Aunque realmente no solamente influye la cámara, sino también la iluminación, los reflejos,...

Para estimar la varianza tenemos dos opciones: utilizar la ecuación estáticamente o dinámicamente. En el desarrollo de este PFC se eligió la manera estática.

Para estimar la varianza de manera dinámica (se asume media cero) se va estimando "adaptativamente" en "regiones homogéneas" de la imagen (el ruido puede afectar de forma distinta a zonas claras/oscuras, cercanas/lejanas,...) cada cierto intervalo de tiempo.

Para estimar la varianza de manera estática hemos seguido el siguiente proceso:

- Primeramente generamos imágenes totalmente negras en la cámara, para ello tapamos el objetivo de la cámara hasta obtener imágenes con los píxeles lo mas parecido a negro (valor 0). En el caso de que solo dispongamos de la secuencia y no podamos obtener las imágenes, tendremos que realizar la misma operación pero con secciones homogéneas de la secuencia (paredes, suelos,…)

- Usando las cámaras instaladas en la EPS conseguimos realizar el experimento. Evidentemente no obtendremos valor 0 debido al ruido procedente de la cámara, también es posible que aparezca algún efecto de offset en los valores que recibimos en las imágenes de la cámara con lo cual nos aparecerán valores que no sólo dependen del ruido (aunque para el cálculo de la varianza nos da lo mismo dicho efecto de offset).

- Para el cálculo efectivo de la varianza se ha generado un pequeño script en Matlab que trabaja sobre un número variable de imágenes obtenidas en el proceso anterior (se generaron 55 imágenes correctas).

Los resultados que se obtuvieron fueron los siguientes:

Imágenes prueba	Varianza media
1	4.1076
5	3.99
10	3.43
15	3.8497
20	4.01
25	4.08
30	4.14
35	4.2178
40	4.291
45	4.3201

Tabla 4-2. Calculo de varianza del ruido introducido por la cámara de la EPS.

NOTA: los resultados son independientes del offset aplicado (aunque en el experimento se escogió un offset de 5)

El fichero donde se encuentra todo el código que ha generado los resultados es "*calculoEstadisticos.m*" .

4.1.2.4 Ventana de aplicación del algoritmo

Observando la fórmula vemos que hay dos parámetros más: q y $g(i,j)^2$ **.** Ambos se referirán a los píxeles donde aplicamos el algoritmo.

El algoritmo se aplicará sobre un píxel, pero para decidir si dicho píxel pertenece al movimiento de un objeto o no tendremos que fijarnos en los píxeles próximos a él (ventana de aplicación).

Inicialmente tenemos una ventana de pxp (5x5 en nuestro sistema) centrada en el píxel a tomar la decisión.

		(i,j)		

- El parámetro *g(i,j)2* será la suma al cuadrado de los píxeles resultantes de una operación entre cuadros. Dicha operación será la de restar el cuadro actual menos el cuadro que representa al fondo.

- El parámetro *q* será el tamaño (en píxeles) de la ventana donde se realizará el análisis. En este caso su valor sería 25.

4.1.2.5 Pruebas

Para probar este algoritmo nos interesará disponer de secuencias donde aparezcan detalles de todos los tipos. Para ello disponemos de la secuencia:

- Secuencia Hall Monitor (300 imágenes de dimensiones 352x240 en formato BMP)

Primeramente calcularemos el ruido para dichas secuencias y luego iremos probando empíricamente los valores del parámetro alfa (umbral) hasta ajustar la detección de movimiento.

Para calcular el ruido se hace tal y como se vio en el apartado 4.1.2.3.

La aplicación se compilará en un archivo ejecutable (*segment.exe*) que tendrá el siguiente formato:

```
   segment <tipoParam> <NumImagenes> <sizeX> <sizeY> <varRuido> <alfa> <umbral_mvto>
<TAF> <TAO> <cuadros_estatico> <umbral_fondo> <ventana_aplicacion>
```

- *tipoParam* → este parámetro nos indicara si la introducción de parámetros al segmentador es de manera reducida (“0”) o extendida (“1”).
- *NumImagenes* → número de imágenes de las que constan las secuencias
- *SizeX* → tamaño en horizontal de las imágenes de la secuencia
- *SizeY* → tamaño en vertical de las imágenes de la secuencia

- *varRuido* → Varianza del ruido que existe en la secuencia de imágenes
- *alfa* → parámetro de la actualización de fondo
- *umbral_mvto* → umbral para la detección de movimiento según el algoritmo.
- *TAF (tiempo actualización de fondo)* → tiempo en cual generaremos un fondo
- *TAO (tiempo actualización de objeto)* → tiempo en el cual realizaremos un análisis del movimiento de objetos en la imagen.
- Parámetros extendidos (disponibles solo si *tipoParam* es igual a 1)
    - *cuadros_estatico* → número de imágenes en las que deberá estar un objeto estático para que pase a formar parte del fondo (por defecto "200")
    - *umbral_fondo* → umbral para una posible detección de movimiento (por defecto "20")
    - *Ventana_aplicacion* → valor q (qxq píxeles) del tamaño de la ventana de aplicación (por defecto "5")

Las secuencias de imágenes de prueba deberán ir en una carpeta llamada "*code/prueba/*", dentro de esa carpeta estarán las imágenes y dos carpetas que tendremos que crear para almacenar los datos de salida, estas son: "*background*" y "*objects*". En ellas se almacenarán respectivamente los fondos generados y la segmentación de objetos generados.

Primeramente usaremos la secuencia Hall Monitor para las pruebas.

Teniendo en cuenta todas las consideraciones, ejecutamos el siguiente comando:

segment 0 300 352 240 15 0.05 1.0e-5 1 1

Obtenemos como resultado una secuencia de backgrounds y de máscaras de nos indican los objetos en movimiento. Estas se generaran con cada nueva imagen de la secuencia que leemos.

A continuación se exponen varios resultados del algoritmo:

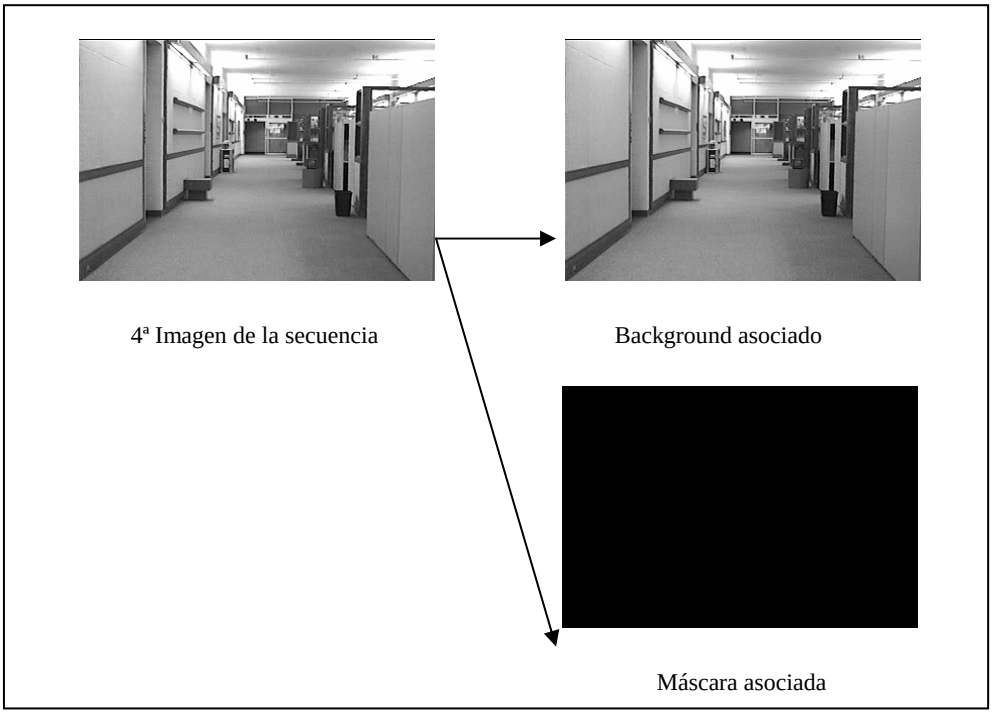


Figura 4-10: Resultados obtenidos tras la segmentación del cuadro 4. No se observa ningún movimiento en la secuencia

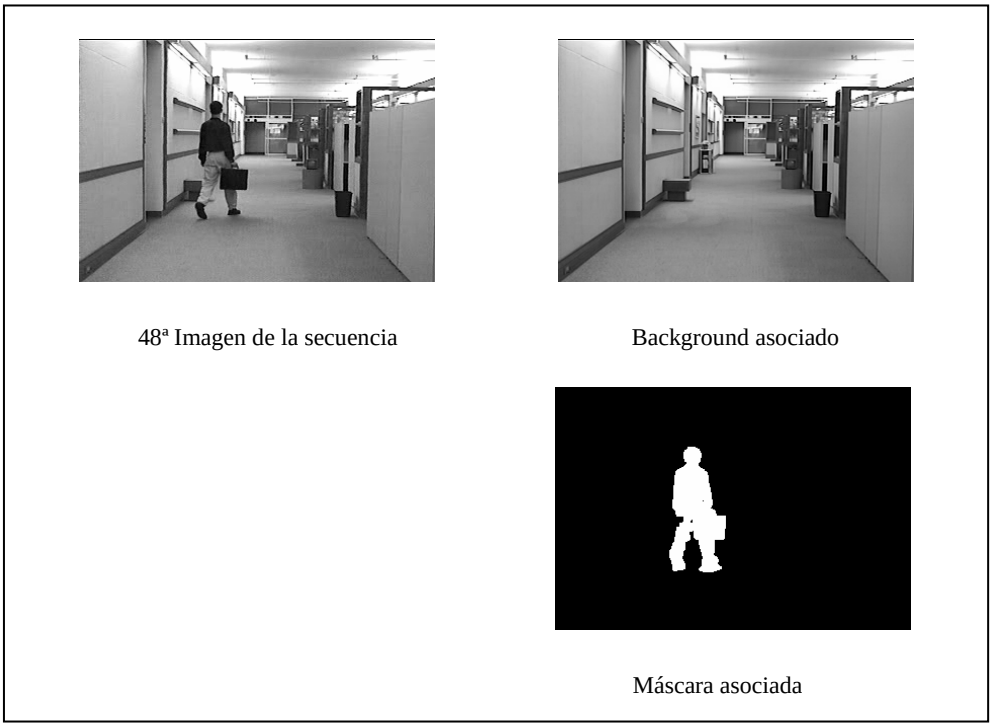


Figura 4-11: Resultados obtenidos tras la segmentación del cuadro 48**.** Se observa el movimiento de una persona y es detectado

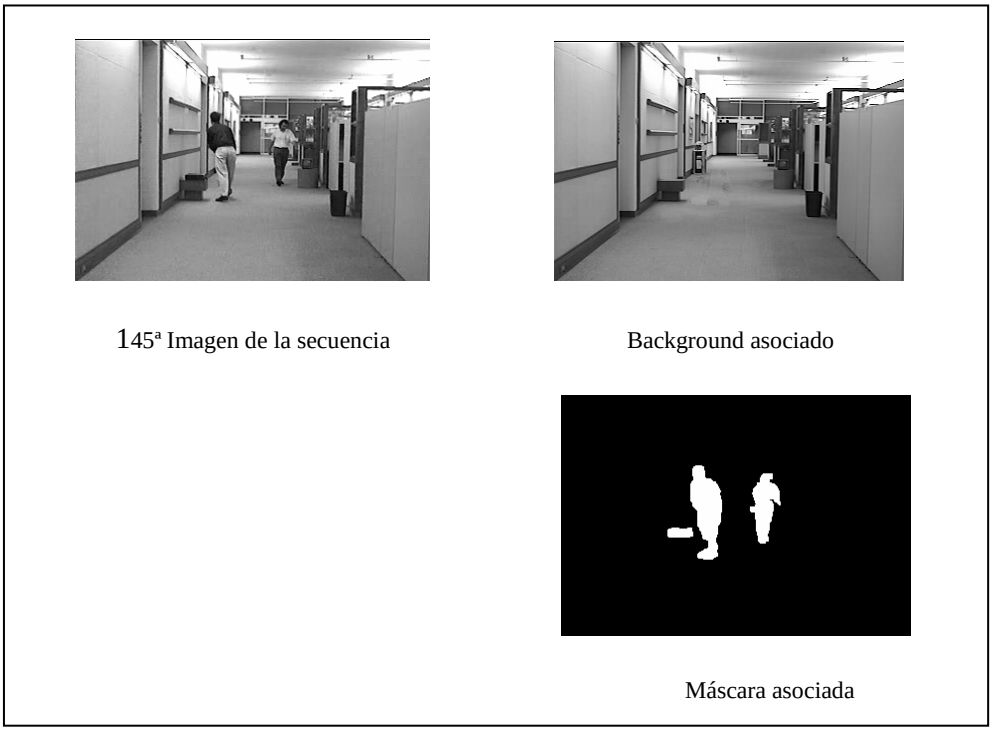


Figura 4-12: Resultados obtenidos tras la segmentación del cuadro 145. . Se observa el movimiento de dos personas y ambas son detectadas. El maletín que es dejado pasara a formar parte del fondo cuando transcurra una número de cuadros en el que el objeto no se mueva (parámetro *cuadros_estatico*)

4.1.3 Seguimiento de objetos

4.1.3.1 Introduccion

En la implementación de este módulo se desarrollarán las siguientes tareas, éstas son:

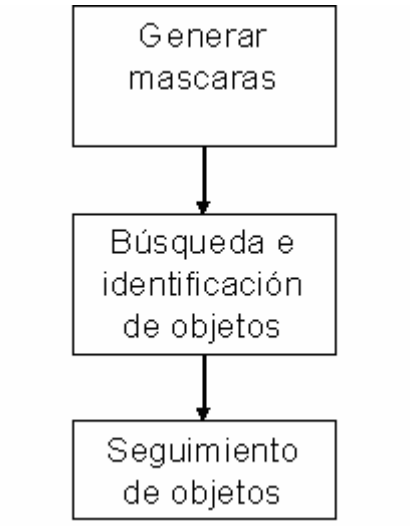


Figura 4-13. Módulo de seguimiento de objetos

Los datos de entrada del módulo son las máscaras (imágenes en formato BMP) generadas por el segmentador de objetos móviles. El seguimiento que se realice será durante un determinado periodo de tiempo determinado por el *Tiempo Actualización de Fondo* (TAF), es decir analizaremos el movimiento de los objetos entre cada actualización de fondo. La frecuencia con que se leerán mascaras dependerá de cada TAO (Tiempo Actualización de Objeto), que nos indicara cada cuanto tiempo genera un análisis de objetos el segmentador.

La salida de datos se generará cada TAF y consistirá en estructuras de información (metadatos/descripciones) e imágenes asociadas. Para cada objeto generaremos una estructura de información (con su id, su trayectoria, sus dimensiones,…) y una imagen que contenga dicho objeto.

Adicionalmente a la salida podremos elegir si queremos que se guarde en un fichero de texto toda la información asociada al movimiento que se ha generado en la ejecución del módulo.

A continuación se explicará cada tarea a realizar y la algoritmia usada en cada una

4.1.3.2 Algoritmo

En este apartado se procede a explicar los algoritmos seguidos en cada tarea a realizar:

- **Generación de máscaras poligonales**

  El algoritmo que se sigue en esta tarea es el siguiente:

  - **Cada Tiempo de Actualización de Objeto (TAO)**
    - **Leer máscara proveniente del módulo de segmentación**
    - **Identificar el número de objetos de la máscara**
    - **Extraer las coordenadas de cada objeto (solo coordenadas máximas y mínimas)**
    - **Generar una nueva máscara con los objetos cuadrados**

  Esta tarea se realiza para luego después poder realizar el seguimiento de los objetos cuadrados en vez de los objetos que tenemos en las máscaras iniciales.

- **Búsqueda e identificación de objetos**

Este módulo buscará los objetos cuadrados que encontremos dentro de cada máscara y los asociará con el objeto anterior.

Este sub-módulo procederá de la siguiente manera:

- **Cada Tiempo de Actualización de Objeto (TAO)**
  - **Identificar los objetos en la mascara cuadrada**
  - **Extraer sus coordenadas (máximas y mínimas)**
  - **Buscar para cada objeto una posible asociación con un objeto perteneciente a la máscara anterior.**

A la hora de desarrollar esta parte, también se pensó en la posibilidad de que la asociación se hiciese directamente con todo el grupo de máscaras que obtenemos cada TAF (*Tiempo de Actualización de Fondo*). Finalmente no se optó por esta solución ya que nos implica un retardo de proceso bastante grande, pues para realizar el seguimiento de un objeto tendríamos que esperar a recibir todas las máscaras y entonces comenzar el proceso de seguimiento. Con la solución actual, la tarea del seguimiento de los objetos la vamos realizando parcialmente (entre dos máscaras consecutivas) mientras vamos recibiendo nuevos datos, con lo cual ahorramos tiempo de proceso en el seguimiento final de cada objeto.

La asociación de podemos realizarla de muchas maneras y con múltiples reglas (distancia euclídea menor, similar tamaño,…). En esta aplicación se ha tomado como característica principal para el seguimiento la distancia euclídea. Para ello calcularemos el centro de masas del objeto rectangular y lo asociaremos con el objeto de la máscara anterior que tenga su centro de masas más cercano al objeto actual.

Esta asociación se producirá siempre que el objeto sea válido. La validez de un objeto la podemos caracterizar de muchas maneras. Por ejemplo, objetos con un tamaño pequeño no serán validos (con esto evitaríamos los errores cometidos en la detección de objetos del módulo anterior), objetos con determinada forma, que los objetos a asociar tengan similar tamaño,… Existen múltiples reglas por las que podríamos descartar un objeto en nuestro análisis.

Adicionalmente para la identificación de objetos cuadrados se desarrolló una función que buscaba regiones conexas de puntos y las marcaba con un valor. Mediante esta función conseguimos identificar objetos (cuadrados o no) que se han podido formar al realizar máscaras rectangulares (unión de objetos). A la hora de realizar las pruebas, se detectó que dicha función agotaba rápidamente la memoria de la pila del sistema. Así pues si el objeto era suficientemente grande podría llegar a producir una excepción en el programa de seguimiento.

Para las pruebas realizadas no se observó dicha situación, pero en las pruebas preliminares con regiones de un tamaño mayor si que se observó.

- **Seguimiento de objetos**

Una vez completado el procesado de todas las máscaras durante un TAF, se procederá al seguimiento (tracking) de los objetos de interés a través de toda la secuencia de máscaras.

Debido a las tareas realizadas anteriormente, este sub-módulo únicamente tendrá que recorrer las estructuras de asociación de movimiento generadas en el proceso anterior.

También en este apartado decidiremos qué objetos son de interés para nuestra aplicación. Se pueden establecer numerosas reglas de decisión para ver si un objeto es válido o no. Inicialmente solo aquellos objetos que tengan más de un determinado tamaño serán considerados.

Podemos considerar reglas como duración mínima, tamaño mínima del objeto, que no se produzca un cambio brusco de tamaño en el seguimiento del objeto,….

Adicionalmente a estas tareas, se genera una imagen por cada objeto válido. Dicha imagen será de las dimensiones que tiene el objeto, en formato BMP y en escala de grises (8bpp).

4.1.3.3 Pruebas

Para las pruebas de seguimiento de un objeto, vamos a realizar lo siguiente:

La primera secuencia consistirá en los 100 primeros cuadros de la secuencia Hall Monitor. En dicha secuencia aparecen errores en la segmentación de objetos (que tendrán que ser descartados por este módulo) y dos objetos que describen una trayectoria a través de la secuencia (sin cruzarse). Los resultados fueron los siguientes:

Figura 4-14. Trayectoria descrita por el objeto identificado (primeros 100 cuadros)

4.1.4 Extracción de características

La tarea de este módulo quedó bien definida en el apartado de diseño.

Conforme a los descriptores de metadatos explicados en la sección 3.2 extraeremos las características necesarias para escribir los descriptores en MPEG-7.

Las características que se extraen son las siguientes:

- Duración del movimiento de cada objeto
- Imágenes asociadas a los objetos detectados (según nivel de transmisión)
- Luminancia dominante de los objetos detectados
- Imagen del fondo calculado
- Luminancia dominante de la imagen de fondo asociada
- Tiempo de creación de cada ítem

4.1.5 Generación de descripciones MPEG-7

Después de la extracción de características se creará un fichero MPEG-7 en el que se guardará parte de la información contenida en los descriptores de metadatos del programa.

Este módulo generará el descriptor maestro que contendrá todos los datos que hemos generado con la aplicación.

El programa desarrollado recibirá descripciones de características generales, características de los objetos y características del fondo. Posteriormente abrirá un fichero (que dependerá del nivel de descripción que queramos obtener) y escribirá en el todas las descripciones de las características en formato MPEG-7.

La salida de este módulo será un fichero con extensión mp7 que se generará cada *Tiempo Actualización Fondo* (TAF).

Adicionalmente al descriptor maestro (de nivel 1), este módulo también generara los otros dos niveles de transmisión disponibles (niveles 2 y 3). Estas opciones serán definidas por el usuario

4.2 Desarrollo de la aplicación de recepción

Al igual que en el capítulo de diseño se hizo un desarrollo modular de la aplicación de recepción, a la hora del desarrollo de dichos módulos también se hará de manera modular. En el apartado 5 se analizará todo el sistema en conjunto y su comportamiento.

La aplicación receptora tendrá unos parámetros de configuración para las secuencias que reconstruirá y estos serán: Tiempo de Actualización de Fondo, Tiempo de Actualización de Objeto y Dimensiones de la secuencia de salida. Realmente las dimensiones de las imágenes no sería un parámetro necesario pues podríamos obtenerlas a partir de los ficheros de descripciones MPEG-7 pero se supone que es un dato conocido a priori (por el usuario) y que así nos ahorraría tiempo de proceso (al no tener que buscar la dimensión dentro del fichero correspondiente).

Después de introducir los parámetros correspondientes, la aplicación funcionará de la manera siguiente:

- **Mientras existan descriptores de secuencias de imágenes**
  - **Leer descriptor de la secuencia de imágenes**
  - **Reconstruir la secuencia de imágenes**
- **Generar un video con todas las secuencias**

4.2.1 Módulo de Lectura de descriptores

Para implementar la tarea de dicho módulo se decidió que la mejor manera era ir leyendo secuencialmente cada línea del fichero de texto e ir identificando (mediante comparaciones) a qué campo nos estamos refiriendo con cada línea leída.

Las comparaciones que realizamos son a nivel de cadenas de caracteres. Por ejemplo si queremos identificar la ruta donde se encuentra la imagen de fondo en el disco entonces buscaremos la primera cadena "<MediaUri>" que aparezca en el fichero y los siguientes datos hasta el carácter "<" se corresponderán con dicha ruta.

Para evitar el problema que surgiría si solamente se realizará una comparación con dicha cadena, se hizo necesario el uso de un contador para ir indicando el paso siguiente a realizar (leer ruta de fondo, leer objeto,….).

El esquema de funcionamiento general de la lectura será el siguiente:

- **Abrir el fichero**
- **Leer la ruta de la imagen de fondo de la secuencia**
- **Por cada objeto encontrado (comparación con "Moving Region id")**
  - **Obtener el comienzo y fin del movimiento ("<MediaTimePoint>")**
  - **Obtener la ruta del objeto ("<MediaUri>")**
  - **Obtener las coordenadas del movimiento ("<KeyValue type=\"startPoint\">" y "<KeyValue type=\"firstOrder\">")**
  - **Obtener la luminancia dominante del objeto ("<LuminanceValueIndex>")**

En la búsqueda de objetos, ésta se realizará (tras obtener la ruta de la imagen de fondo) hasta el final del fichero.

4.2.2 Módulo de Generación de secuencia de imágenes

La tarea de generación de la secuencia de imágenes quedó bastante bien definida en el apartado de diseño de la aplicación receptora.

Antes de iniciar la generación de la secuencia, este módulo deberá primero cargar las imágenes de los objetos (que dependerán del nivel de transmisión seleccionado). Así tendremos varias opciones:

- Nivel 1 → en base a la duración del movimiento de cada objeto, calcularemos cuantos análisis de objeto ha estado activo el objeto y cargaremos imágenes desde "*000.bmp*" hasta "*XXX.bmp*" donde XXX se corresponderá con el número anteriormente calculado. La localización de dichas imágenes vendra determinada por el campo <MediaUri> (en el descriptor MPEG-7) asociado a cada objeto.
- Nivel 2 → como en este nivel de reconstrucción solo existirá una imagen. Cargaremos la primera imagen "*000.bmp*" que nos servirá para reconstruir el objeto en la secuencia entera.
- Nivel 3 → no necesitamos cargar imágenes, nos valdrá con la luminancia dominante de cada objeto proporcionada por el módulo anterior

Una vez que tengamos seleccionado el nivel de reconstrucción deseado, las imágenes de los objetos cargadas (dependiendo del nivel) y las descripciones del movimiento de cada objeto, procederemos de la siguiente forma:

- **Generaremos una secuencia de imágenes (dicha secuencia consistirá en "n" imágenes idénticas a la imagen de fondo)**
- **Para cada imagen, identificaremos si tiene movimiento asociado y en caso positivo se procederá a la inserción del objeto.**
    - **Para la inserción del objeto tendremos tres opciones (dependiendo del nivel de reconstrucción)**
        - **Nivel 1 → imagen correspondiente al cuadro actual y al objeto actual (se dispone de una imagen para cada cuadro donde esta presente el objeto)**
        - **Nivel 2 → imagen del objeto durante toda la secuencia**
        - **Nivel 3 → luminancia dominante de las imágenes del objeto durante toda la secuencia**

El parámetro "n" anteriormente descrito tendrá que ver con la frecuencia con la que realizamos el análisis de fondo (*Tiempo de Actualización de Fondo*) y frecuencia de cuadros por segundo (*fps*). La relación será de generar *n=TAF*fps* imágenes (si TAF esta en segundos) para cada secuencia.

En la siguiente figura se muestran los resultados del modulo de generación de secuencias a distintos niveles.

Figura 4-15. Cuadros 50, 100, 150, 200, 250 y 300 de la reconstrucción de nivel 1 de la secuencia Hall Monitor

Figura 4-16. Cuadros 50, 100, 150, 200, 250 y 300 de la reconstrucción de nivel 2 de la secuencia Hall Monitor

Figura 4-17. Cuadros 50, 100, 150, 200, 250 y 300 de la reconstrucción de nivel 3 de la secuencia Hall Monitor

4.2.3 Módulo de presentación

Inicialmente dispondremos de imágenes en formato BMP y en escala de grises y conforme a las especificaciones del diseño, tendremos que obtener un video en MPEG-2 que contenga la secuencia reconstruida por nuestro sistema.

Para generar el video usaremos la herramienta de creación de vídeo ***ffmpeg***[13]. Dicha herramienta nos permitirá crear un video a partir de una secuencia de imágenes en formato JPEG (24bpp). La ventaja de esta herramienta es que la podremos usar desde la línea de comando con lo que podremos integrarla en nuestro módulo de presentación.

Como inicialmente partimos de una secuencia de imágenes en formato BMP, necesitaremos otro programa que nos realice una conversión de BMP de 8 bpp a JPEG (24 bpp). Dicho programa deberá ser también posible ejecutarlo desde línea de comando para permitir una integración con nuestro sistema.

De todas las herramientas disponibles se decidió por un programa de código libre ***bmptojpg.exe*** que nos permite cambiar una imagen de formato BMP a JPEG (incluso variando la calidad de la imagen transcodificada).

Así pues el módulo de presentación recibirá como parámetro el nombre del video a generar y la secuencia de imágenes en BMP, las transcodificará con el programa *bmptojpg.exe* (sin pérdidas) y posteriormente generará el video con la herramienta *ffmpeg.exe.*

Más concretamente los comandos que utilizaremos para los distintos programas serán los siguientes:

- Transcodificar todas las imágenes en formato BMP (8 bpp) a JPEG (24 bpp) sin pérdidas:

```
bmptojpg.exe 100 *.bmp
```

[13]Para más información consulte la página Web http://ffmpeg.mplayerhq.hu/

- Creación del video con imágenes en formato JPEG (24 bpp) y con la nomenclatura “imageXXX.jpg”

```
ffmpeg.exe –r <fps> -i image%03d.jpg  <nombre_video>
```

Donde el parámetro *fps* nos indicará a que frecuencia de cuadro queremos generar el vídeo.

5 Integración, pruebas y resultados

En esta sección se presentará la integración de las dos aplicaciones desarrolladas (transmisora y receptora). Posteriormente se irán explicando los distintos problemas que han ido surgiendo mientras se integraban los distintos módulos.

Por último se realizarán pruebas con distintas secuencias de imágenes para comprobar los resultados obtenidos con el sistema completo.

5.1 Sistema completo

5.1.1 Sistema transmisión

5.1.1.1 Integración de módulos

El desarrollo de la aplicación transmisora se ha llevado a cabo en módulos, que se han ido desarrollando por separado. Esta decisión fue tomada para ir comprobando que su funcionamiento era el correcto.

Podemos distinguir los siguientes módulos:

- Módulo de generación de background
- Módulo de estimación de movimiento
- Módulo de seguimiento de objetos
    - Identificación de objetos (regiones de interés conexas)
    - Asociación de objetos entre cuadros consecutivos
    - Seguimiento de objetos en la secuencia entera
- Módulo de extracción de características
- Módulo de generación de descriptores
    - Nivel 1
    - Nivel 2
    - Nivel 3
- Módulo de transmisión

Posteriormente al desarrollo de los módulos por separado se procedió a su integración. Para la integración se tuvieron en cuenta distintos aspectos.

Los principales datos de entrada de la aplicación son las imágenes, sus dimensiones y los tiempos con los que generaremos los análisis (*Tiempo de Actualización de Fondo* y *de Objeto*).

Como todos los algoritmos diseñados están preparados para trabajar con grupos de imágenes, es decir, el análisis de fondo se realizara cada "m" imágenes leídas o el análisis de objetos se realizara cada "n" imágenes leídas, entonces necesitaremos crear nuevos parámetros para nuestros módulos. Para estos nuevos parámetros necesitaremos que se nos proporcione otro parámetro básico de entrada, la frecuencia de cuadros por segundo (*fps*).

Así pues tendremos dos nuevos parámetros (que calculara internamente la aplicación):

- Frecuencia de Actualización de Background (medida en cuadros)

  FUB = *TAF * fps*

- Frecuencia de Actualización de Objeto (medida en cuadros)

  FUO = *TAO * fps*

Todos los módulos debían tener la posibilidad de ver los resultados que generan, es decir, que si por ejemplo se quiere observar los resultados de la detección de movimiento dentro de la secuencia de imágenes tengamos posibilidad de ver las imágenes correspondientes. Esta característica adicional hace posible detectar en que módulo podrían estar los posibles fallos en un análisis de movimiento erróneo.

Así pues podremos observar información sobre los siguientes aspectos:

- Fondos generados
- Máscaras de detección de movimiento e identificación de objetos cuadrados
- Asociación de movimiento entre cuadros consecutivos

Todos los módulos han sido desarrollados de manera que desarrollan el trabajo de manera secuencial, es decir que si iniciamos el análisis de una secuencia, éste no podrá ser detenido hasta que no termine el proceso. La aplicación que se ha desarrollado esta destinada a funcionar en un entorno durante un determinado tiempo y si por cualquier error se introducen parámetros erróneos, deberíamos esperar a que termine el análisis para poder volver a tener el control del sistema. Para poder tener el control del sistema y así poder detener el proceso de análisis en determinados instantes se hará uso de la tecnología de threads.

La integración de los distintos módulos se fue haciendo de forma gradual, y se fue probando la aplicación con cada módulo añadido hasta obtener todo el sistema de transmisión completo.

5.1.1.2 Entorno visual

Para realizar un entorno de manejo de la aplicación se disponía de dos opciones: por línea de comandos o un entorno visual.

Se eligió desarrollar un entorno visual debido a que es más simple de manejar por el usuario.

A la hora de desarrollar un entorno visual para la aplicación se pensaron en los siguientes requisitos:

- Necesidad de una ventana principal donde introducir los parámetros principales de la aplicación (tamaño de las imágenes, *Tiempo de Actualización de Fondo*, *Tiempo de Actualización de Objeto* y *Frecuencia de cuadros por segundo*)
- Ventanas para introducir los distintos parámetros de los módulos que forman la aplicación. Además una breve explicación de la tarea que desarrolla cada módulo se incluirá en cada pantalla.

Para la tarea del desarrollo de la aplicación visual se disponía de dos opciones: MFC (Microsoft Fundation Classes) y Java.

Finalmente se eligió MFC debido a que todos los algoritmos estaban desarrollados en C++ y su integración con el entorno visual era más sencilla. La principal desventaja que se obtuvo es la portabilidad, nuestra aplicación solo se podrá ejecutar en sistemas con sistema operativo Windows instalado.

Los resultados fueron los siguientes:

- Ventana principal

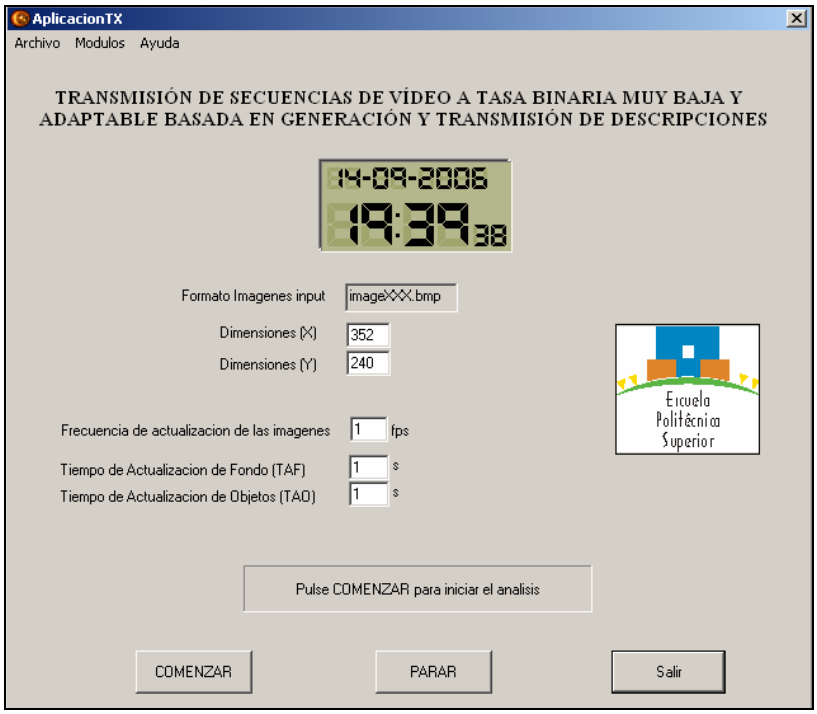


Figura 5-1. Ventana principal del entorno visual de la aplicación de transmisión

Desde dicha ventana podremos introducir los parámetros principales de la aplicación y accederemos a los distintos módulos seleccionando la opción del menú *Módulos->*…

- Configuración de los Módulos

Accediendo a dichos módulos tal y como se explica en el párrafo anterior, nos aparecerá la ventana del correspondiente módulo con los posibles parámetros de configuración y los posibles datos a salvar.

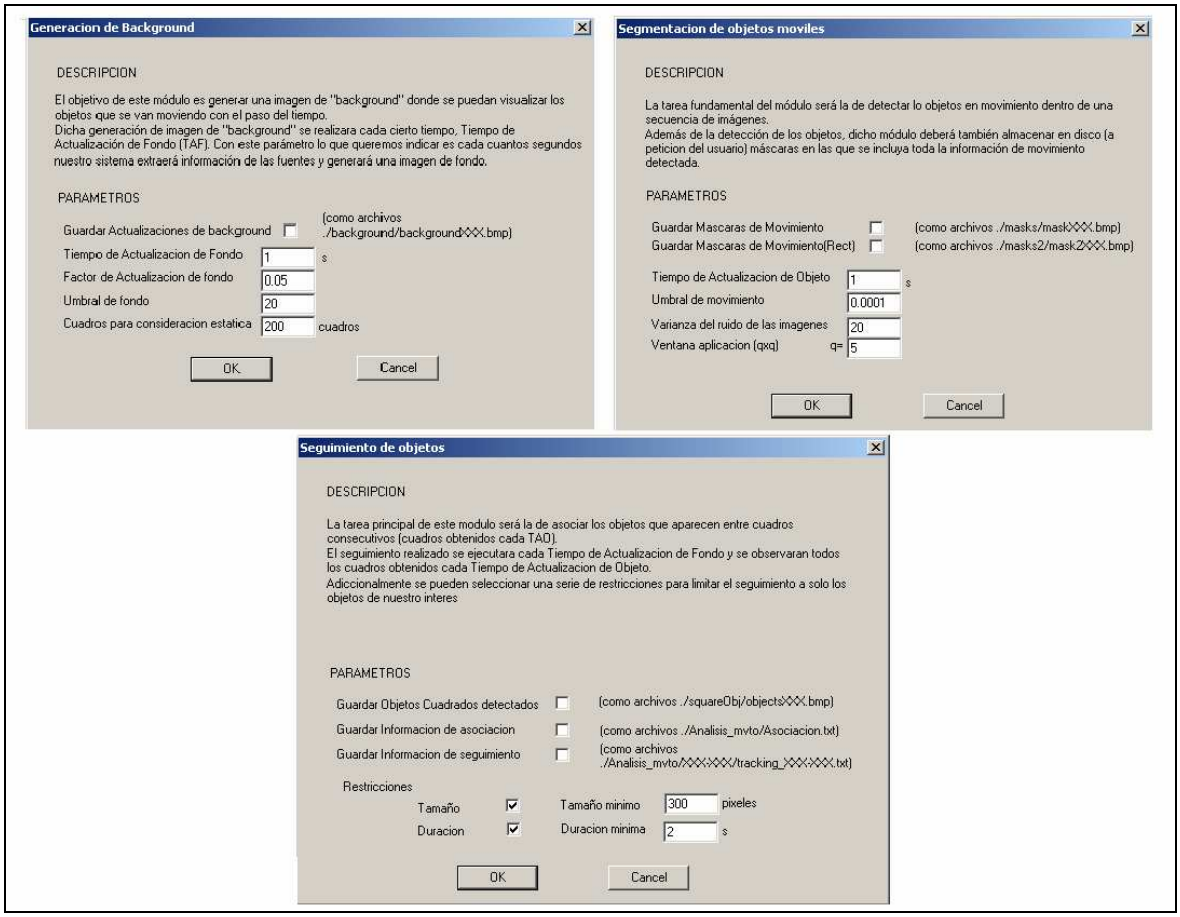


Figura 5-2. Ventanas de configuración de algunos módulos de la aplicación

5.1.2 Sistema de recepción

5.1.2.1 Integración de módulos

El desarrollo de la aplicación receptora se ha llevado a cabo en módulos, que se han ido integrando a medida que se iban desarrollando. Esta decisión fue tomada debido a la simplicidad de las tareas a realizar por dichos módulos

Podemos distinguir los siguientes módulos:

- Módulo de lectura de descripciones
- Módulo de generación de secuencias
    - Nivel 1
    - Nivel 2
    - Nivel 3
- Módulo de creación de video

A la hora de integrar todos los módulos, se definieron como parámetros principales de la aplicación los siguientes:

- Dimensiones de la secuencia de reconstruir
- *Tiempo de Actualización de Fondo* (TAF)
- *Tiempo de Actualización de Objeto* (TAO)
- Frecuencia de cuadros por segundos para la generación del vídeo (fps)
- Nivel de reconstrucción deseado

Igual que en la aplicación transmisora, los parámetros TAF y TAO se han de modificar para generar la secuencia de imágenes. Las nuevas variables son las mismas:

- Frecuencia de Actualización de Background (medida en cuadros)

  FUB = *TAF * fps*

- Frecuencia de Actualización de Objeto (medida en cuadros)

  FUO = *TAO * fps*

Con respecto a su funcionamiento, la aplicación irá ejecutando secuencialmente los módulos de "Lectura de Descripciones" y de "Generación de Secuencias" hasta que no encuentre más descripciones. Todas las secuencias generadas con cada análisis de secuencia (descriptor, imagen de fondo y posibles objetos de imagen) irán siendo almacenadas hasta que no se encuentren más descripciones. En ese instante se realizará una llamada al módulo de "Creación de Vídeo", que creará un vídeo con todas las secuencias generadas anteriormente.

Para la creación del video se usarán las secuencias de imágenes y el parámetro "frecuencia de cuadros por segundo".

Adicionalmente a los módulos desarrollados, se pensó que sería buena idea poder visualizar los resultados obtenidos en la reconstrucción del algoritmo (niveles 1, 2 y 3). Para ello se programaron tres aplicaciones sencillas cuya función era la de visualizar los videos reconstruidos en cada nivel. Dichas aplicaciones fueron integradas con la aplicación receptora.

5.1.2.2 Entorno visual

Para el entorno visual de esta aplicación se eligió el mismo formato que para la aplicación transmisora, entorno visual con sistema de ventanas MFC.

La aplicación consta de una ventana principal

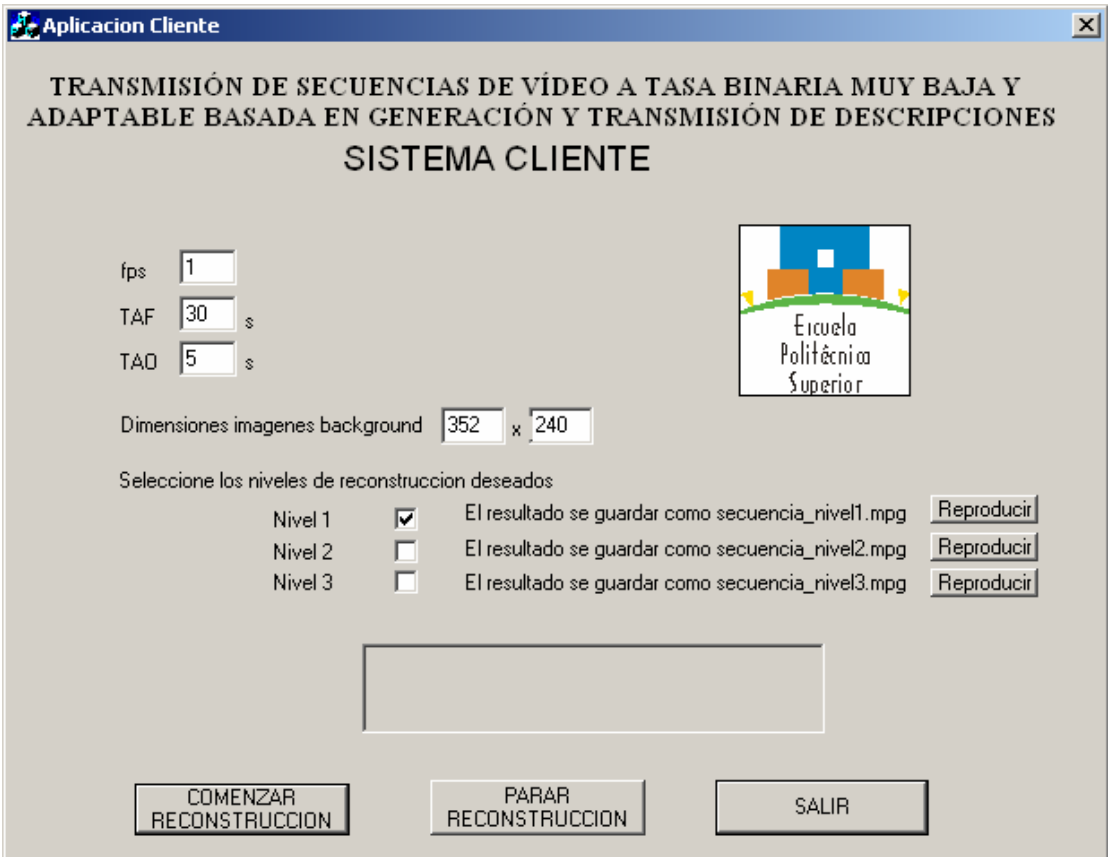


Figura 5-3. Ventana principal de la aplicación cliente

En la que podremos introducir los parámetros principales para la reconstrucción. Adicionalmente se podrá seleccionar el nivel de reconstrucción que queremos obtener.

Para la posterior visualización de los resultados, solamente debemos pinchar en el botón correspondiente y no saltara una ventana como la siguiente:

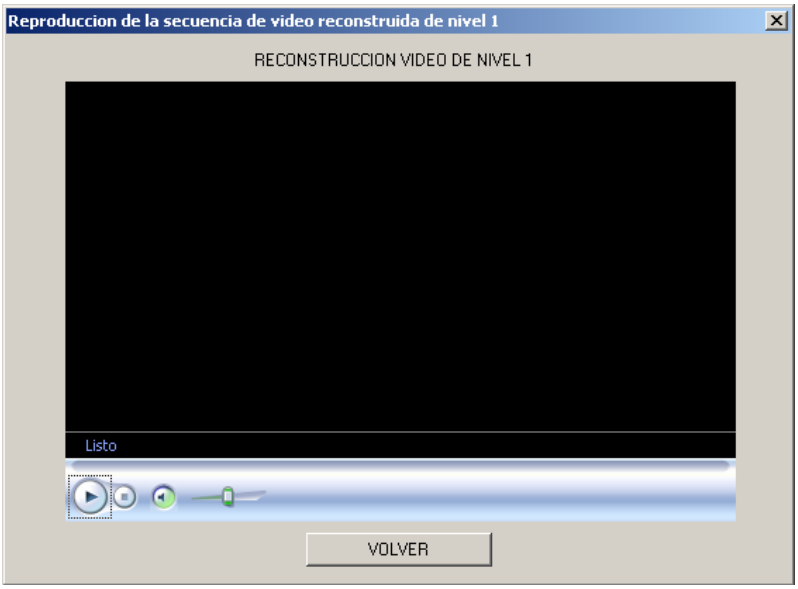


Figura 5-4. Ventana de visualización de la reconstrucción de la secuencia (nivel 1)

Donde para observar el vídeo simplemente tendremos que pulsar el botón 

5.1.3 Pruebas y resultados

Ambas aplicaciones se probaron en varias máquinas. La única diferencia apreciable entre unas y otras es la mayor o menor rapidez del sistema al ejecutarse el proceso de análisis de las secuencias.

Para una máquina con las siguientes características:

- Procesador *INTEL PENTIUM 4 a* 3.25 GHz.
- *512 MB* de *RAM.*
- *120 GB* de disco duro.

Las pruebas que realizaremos serán con la siguiente secuencia de imágenes:

- Secuencia Hall Monitor (300 imágenes de dimensiones 352x240 en formato BMP)

A la hora de realizar las pruebas nos interesará principalmente el tiempo que tarda en ejecutarse el análisis, el ahorro de información enviada y la calidad de la información reconstruida.

Para realizar las pruebas se necesitará que las imágenes de entrada se dispongan en una carpeta llamada “images” que colgara del mismo directorio en el que este la aplicación. En dicha carpeta se deberán encontrar todas las imágenes de la secuencia con el nombre “imageXXX.bmp”. Deberán estar en formato BMP y en escala de grises (8bpp).

Los resultados que producirá el programa se almacenarán en una carpeta llamada “output”. Dentro de ella se encontrará toda la información que podremos extraer del proceso y la información a transmitir dependiendo del nivel que hayamos seleccionado.

La información a transmitir colgará de la carpeta “tx”. La carpeta “tx” contendrá tres carpetas que se generarán en función del nivel o niveles de transmisión solicitados (Nivel 1, 2 ó 3)

5.1.3.1 Sistema completo

Como primera prueba se analizaron la secuencia Hall Monitor. El tamaño inicial de las 300 imágenes es 25,7 MB. Ejecutamos el programa y le ajustamos los siguientes parámetros:

- Fps =1
- TAF =30
- TAO =5
- Niveles de transmisión seleccionados = 1,2 y 3

El proceso tardó en ejecutarse 13 segundos. Es un tiempo lejos de una aplicación en tiempo real, pues una secuencia de vídeo a una frecuencia normal (30 fps) generaría 300 imágenes en 10 segundos. Aunque hay que tener en cuenta que se han generado los tres niveles de descripciones.

Para comprobar la efectividad de la aplicación en tiempo real, realizamos la misma prueba pero generando solo un nivel de descripción. Cuando se generó el primer nivel de descripción se tardaron 12 segundos. Con el segundo nivel de descripción el proceso se completó en 11 segundos y el tercer nivel se completo en 10 segundos.

Volviendo a la primera prueba, el resultado que obtuvimos fue un análisis de la secuencia cada 30 cuadros para obtener una actualización de fondo y cada 5 cuadros para obtener una actualización de objeto. Automáticamente se generó la carpeta “tx” y su contenido era:

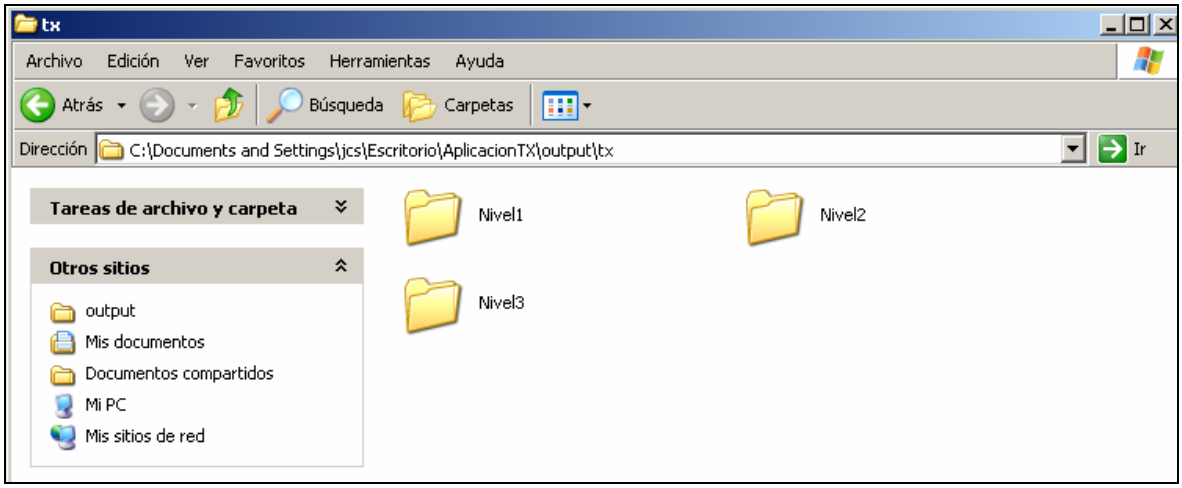


Figura 5-5. Contenido de la carpeta “tx” donde se almacenan los datos a transmitir

Dentro de cada carpeta encontramos un análisis por cada 30 cuadros.

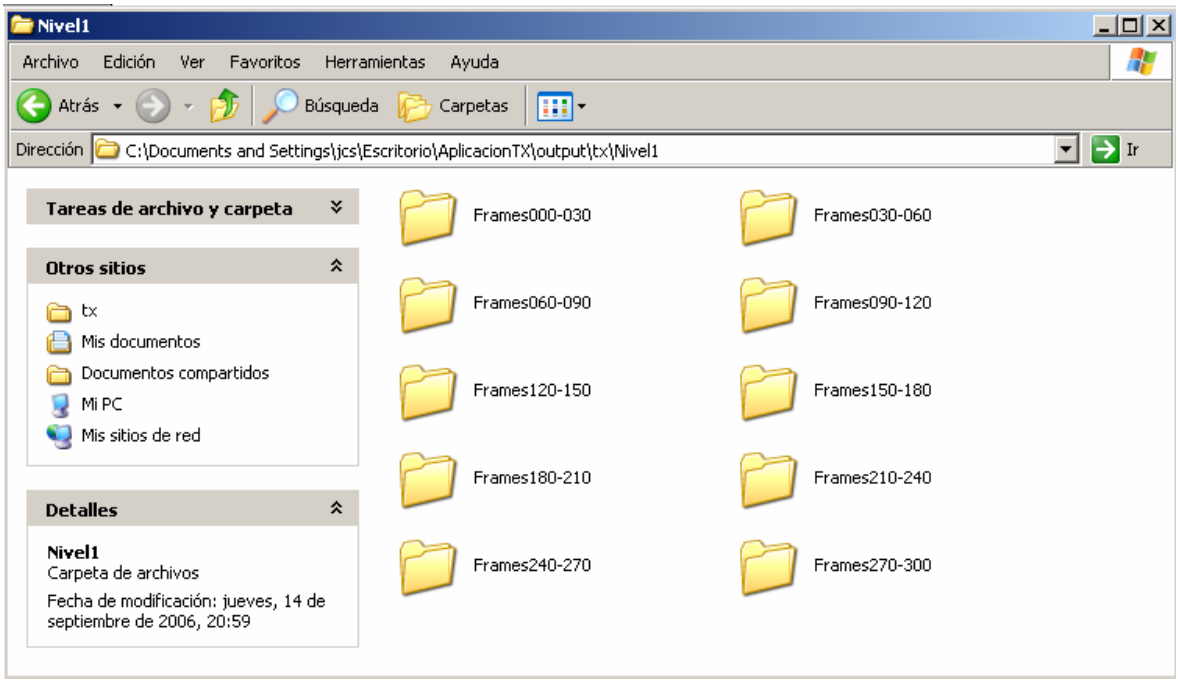


Figura 5-6. Contenido de la carpeta "tx/Nivel 1"donde se almacenan los datos a transmitir del nivel 1

Y por cada análisis tenemos un descriptor de movimiento, una imagen de background e imágenes de objetos (dependiendo del nivel de transmisión).

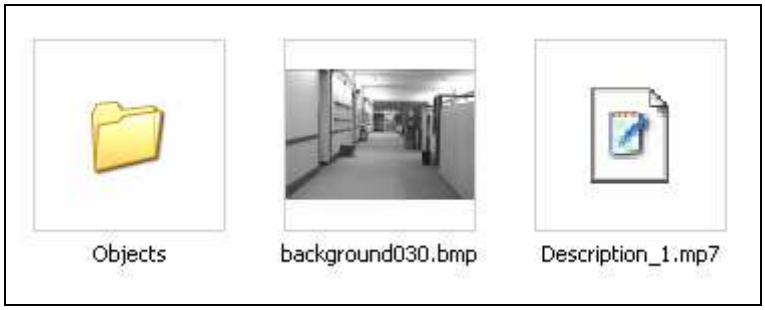


Figura 5-7. Contenido de la carpeta "tx/Nivel1/Frames000-030", donde se analizan de los cuadros 000 a 030 de la secuencia

Para más información sobre la información generada y otras pruebas, consultar el anexo "**J RESULTADOS DEL ANALISIS DE UNA SECUENCIA DE IMAGENES**" donde se detallará toda la información generada.

El tamaño inicial de la información (300 imágenes) era de 25,7 MB y el tamaño de la información obtenida tras el proceso de nuestra aplicación fue el siguiente:

- Nivel 1 → 1,41MB
- Nivel 2 → 0.98 MB

- Nivel 3 → 0.93 MB

Con lo cual obtenemos una reducción del tamaño de la información que oscila entre el 96% (con el nivel 3) y el 94% (con el nivel 1). Para ver si la reducción del tamaño no implica una pérdida significativa de calidad, procedemos a ejecutar la aplicación de reconstrucción y a observar los resultados.

Para ejecutar la aplicación de recepción basta con copiar la carpeta "tx" al directorio donde se encuentre la aplicación receptora y renombrar la carpeta a "rx".

Adicionalmente a la aplicación de recepción y con el fin de generar el vídeo correspondiente a la secuencia reconstruida, necesitaremos los programas "ffmpeg.exe" y "bmptojpg.exe". Para el funcionamiento de la aplicación "ffmpeg" necesita de las librerías "AVSredirect.dll" y "pthreadGC2.dll".

A la hora de reconstruir introducimos los mismos parámetros que en el transmisor y ejecutamos la aplicación. El proceso tardó 23 segundos en ejecutarse.

El motivo de que el proceso sea tan lento es debido a la escritura de las imágenes en formato BMP y su posterior transcodificación mediante el programa externo "bmptojpg.exe". Además para la creación del video, el programa "ffmpeg.exe" también nos introduce un retardo considerable.

Para probar si la aplicación podría funcionar en un entorno de tiempo real, probamos a generar solo un nivel de descripción (concretamente el nivel 1) y tardó 11,5 segundos en ejecutarse todo el proceso.

Como ejemplo representativo de los resultados se incluye la siguiente figura (cuadros extraídos de las secuencias de vídeo):

Imagen original Imagen reconstruida (Nivel 1)

Imagen reconstruida (Nivel 2) Imagen reconstruida (Nivel 3)

Figura 5-8: Niveles de reconstrucción obtenidos en el receptor

Donde se puede observar que la calidad que obtenemos en la imagen reconstruida en el nivel 1 se aproxima con bastante exactitud a la secuencia original. Mientras que los niveles 2 y 3 obtienen aproximaciones con bastante menos calidad en relación con la reducción del tamaño de la información.

5.1.3.2 Demostración

Para poder observar todos los niveles de transmisión y poder compararlos con el original se desarrolló una pequeña aplicación que mostrase todos los videos al mismo tiempo.

La aplicación se muestra en la siguiente figura:

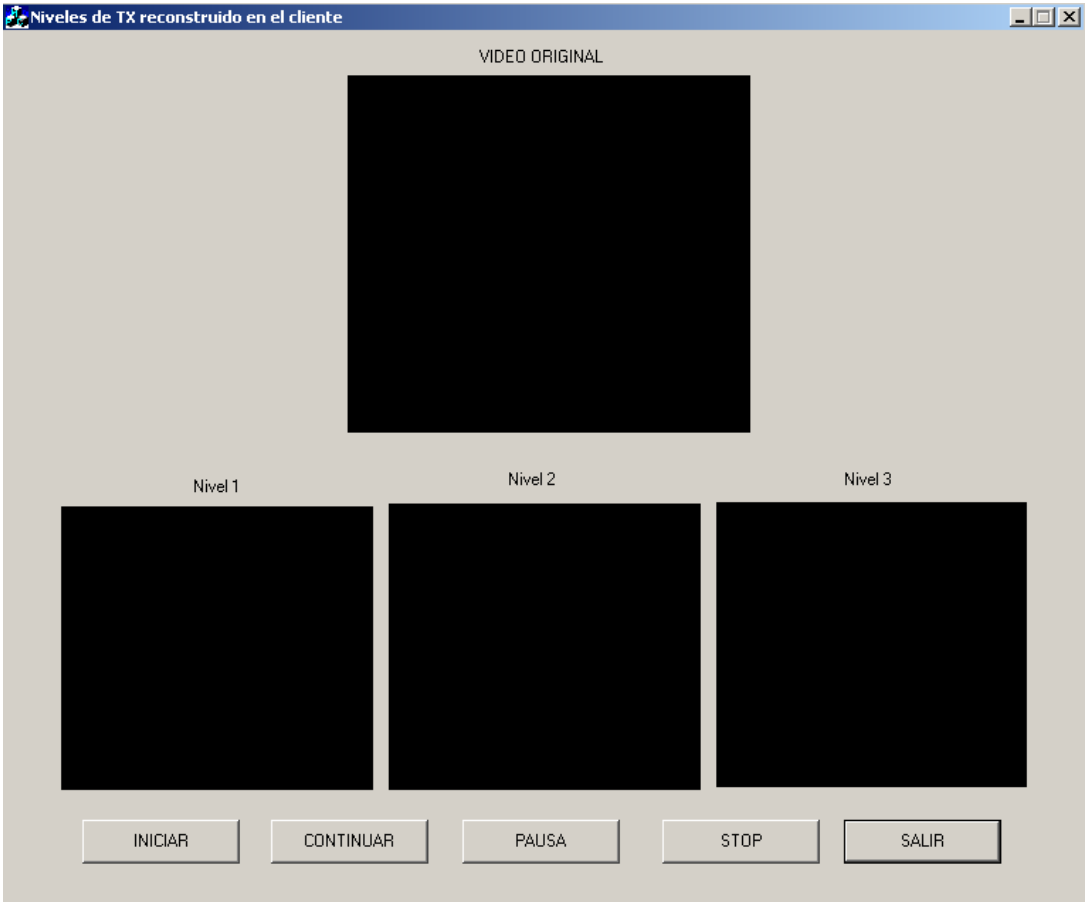


Figura 5-9: Aplicación demo sobre niveles de reconstrucción obtenidos en el receptor

Y para su correcta ejecución necesitara que en su mismo directorio estén presentes los siguientes videos:

- *video.mpg* → video de la secuencia original
- *video1.mpg* → video de la secuencia reconstruida de nivel 1
- *video2.mpg* → video de la secuencia reconstruida de nivel 2
- *video3.mpg* → video de la secuencia reconstruida de nivel 3

6 Conclusiones y trabajo futuro

6.1 Conclusiones

En general, el sistema desarrollado cumple con los objetivos propuestos, si bien los programas de demostración se limitan a terminales con sistema operativo Windows (esto último impuesto por la aplicación Visual desarrollada con tecnología MFC) y por lo tanto no proporcionan realmente un acceso universal (sin embargo los algrotimos desarrollados son susceptibles de ejecutarse en cualquier terminal).

Inicialmente se buscaba un sistema que consiguiera reducir la tasa binaria.

La principal ventaja del sistema desarrollado no es su tiempo de ejecución sino el ahorro obtenido en información generada. Tal y como se comento en el apartado 5.2.3.1, se obtiene una reducción entorno al 95% (que dependerá de la secuencia analizada, los objetos identificados en ella y el nivel de descripción seleccionado). Con este ahorro de tamaño, realizaremos una transmisión a una tasa binaria más baja. Al reducir la tasa, estaremos reduciendo el uso de recursos y consecuentemente el gasto en ellos.

Respecto a la posible operación en tiempo real, en el nivel de mayor calidad (Nivel 1), la aplicación transmisora tardaba 12 segundos en generar toda la información (que si suponemos que la secuencia la componen imágenes a 30 fps, son 10 segundos de secuencia). A la hora de reconstruir el mismo nivel en la aplicación receptora, se tardó 11,5 segundos en completar la ejecución. Estos tiempos hacen que nuestro sistema se aproxime a la ejecución en tiempo real. Si queremos que nuestro algoritmo trabaje en tiempo real, solamente deberemos ajustar los parámetros de análisis (TAF y TAO) para que se produzca un análisis acorde a la frecuencia de generación de imágenes (fps). Actualmente se analiza una secuencia de manera completa, en cambio si se analizara y transmitiese cada TAF, se podría lograr enviar cada TAF con un retardo mucho menor, de forma que podríamos acercarnos mucho mas a tiempo real. Otro inconveniente es la optimización del código, ya que el proyecto se trataba de demostrar la viabilidad del sistema. Cabe destacar que los resultados son más que esperanzados para poder implementar dicho concepto en sistemas AD-HOC.

Uno de los problemas en la reconstrucción en tiempo real es la llamada los programas externos que realiza la transcondificación entre BMP y JPG y la generación de video con la secuencia de imágenes en JPG. Dicho problema sería fácilmente solucionable incorporando un visualizador de la secuencia en la propia aplicación. En las pruebas se usaron secuencias de 300 imágenes, para

secuencias con mayor número de imágenes no debería suponer un incremento notable en el tiempo de ejecución (ya que la tarea más costosa es la llamada a los programas.

Uno de los inconvenientes es, como cabría esperar, el gran consumo de recursos empleado por la aplicación transmisora. El hecho de cargar en memoria las imágenes de la secuencia, generar máscaras de movimiento, identificación de objetos,…. El gran número de variables que maneja la aplicación hace que necesite gran cantidad de recursos.. Una posible solución a esto seria una correcta optimización de los algoritmos que procesan las secuencias de imágenes.

El sistema desarrollado pone de manifiesto las posibilidades que ofrecen las tecnologías empleadas para crear sistemas automáticos de vídeo-seguridad.

6.2 Trabajo futuro

Debido al desarrollo modular del sistema, cualquier parte de ambas aplicaciones (transmisora y receptora) podría ser susceptible de ser mejorada. En este proyecto simplemente se han sentado las bases de cómo debería ser el sistema y unos primores prototipos (a modo de pruebas de viabilidad) de cada módulo.

Aún así existen diversas cuestiones de un interés especial que podrían ser líneas de trabajo futuro:

- A nivel de generación de descripciones MPEG-7, surgen dos cuestiones principales
    - **Ampliación con nuevos niveles de transmisión/reconstrucción.** Con el objetivo de reducir la tasa binaria a la que enviar los datos y así poder optimizar el ancho de banda disponible, se sugieren nuevos niveles:
        - **Nivel 4**: descripción MPEG-7 y el fondo bajo demanda. Con el fondo bajo demanda se quiere indicar que el fondo no se enviara siempre, solamente cuando la diferencia con el fondo calculado anteriormente sea significante.
        - **Nivel 5**: eventos que han ocurrido en la secuencia de imágenes. Estos pueden ser: “entra moving object”, “sale moving object”, “nuevo objeto parado”, “objeto parado pasa a movimiento”, “objeto parado pasa a fondo”,…

          Para ello se debería tener una lista de eventos definidos (incluyendo unknow) para identificar los posibles eventos de un objeto.

- **Codificación de los descriptores MPEG-7 a BiM.** Con esta codificación de los descriptores se pretende optimizar el espacio que ocupan en el disco los descriptores. Realizando esta tarea lograríamos unas tasas de transmisión mas bajas. A realizar dicha codificación nos permitiría dar unos datos más "realistas" al tratarse de comprimir los XML a formato binario.

- **Módulo de seguimiento: características avanzadas de seguimiento.** En la línea de conseguir un sistema más robusto, el módulo donde las mejoras serían más necesarias es el de "tracking". El seguimiento desarrollado en el PFC se basa en la distancia euclídea de dos puntos (los centros de masas de los objetos) y tal y como se explicó ese método presenta varios inconvenientes, aunque es bastante sencillo de implementar. El problema surgirá cuando los objetos de la secuencia se crucen entre si, el algoritmo de seguimiento los confundirá y realizará un seguimiento erróneo. La línea de trabajo sería conseguir un módulo más efectivo, que realizase un seguimiento de objetos en base a varios parámetros y que fuese más eficaz que el desarrollado.

- **Sistema de alarmas.** En la línea de hacer el sistema más automático, se propone la realización de un sistema de alarmas. Dicho sistema tendrá la función de avisar al usuario (aplicación receptora) de objetos que se mueven por zonas prohibidas. Este nuevo módulo debería ser totalmente parametrizable para así poder aumentar su funcionalidad y versatilidad.

- **Sistema en tiempo real.** Con el objetivo de hacer un sistema más eficaz y desarrollar un producto lo más vendible posible, se propone modificar ciertas rutinas para que las aplicaciones lleguen a trabajar en tiempo real.

  En la aplicación transmisora, el módulo de seguimiento es el que tiene mayor cálculo computacional. Una posible optimización de sus algoritmos haría que el sistema fuese más rápido en tiempo de ejecución.

  Respecto a la aplicación receptora, poco se puede modificar. Esta bastante optimizada y las tareas que realiza son bastante sencillas. Una posible mejora sería integrar los programas que utiliza (ffmpeg y bmptojpeg) dentro de la misma aplicación. Asi ahorraríamos el tiempo que tarda la aplicación receptora en realizar las llamadas a los programas.

- **Implementar un sistema que analice la calidad de las imágenes reconstruidas.** Con la idea de realizar una reconstrucción con mas calidad y optimizando los recursos, se pretendería diseñar un sistema que comparase la calidad de las secuencias generadas.

- **Parser XML.** Debido a la simplicidad de los descriptores generados en este sistema se desarrolló un módulo de lectura bastante sencillo. En la medida que las descripciones generadas se hagan más complejas, será inviable extender su funcionalidad. Una posible solución (aprovechando las ventajas que nos da XML) sería la realización de un “parser XML”.

Referencias

[1] K.N. Platanioitis, C.S. Regazzoni (eds.), “Special Issue in Visual-centric Surveillance Networks and Services”, IEEE Signal Processing Magazine, 22(2), Marzo 2005.

[2] B.S. Manjunath, P. Salembier, T. Sikora (eds.), “Introduction to MPEG 7: Multimedia Content Description Language,”, John Wiley and Sons, 2002

[3] O. Steiger, T. Ebrahimi, A. Cavallaro, “Real-time generation of annotated video for surveillance”, Proc. of WIAMIS’05, Montreux, Suiza, Abril 2005.

[4] Vittorio Ferrari, Tinne Tuytelaars, Luc Van Gool, “Simultaneous Object Recognition and Segmentation by Image Exploration”, Computer Vision Group (BIWI), ETH Zürich, Switzerland, ESAT-PSI, University of Leuven, Belgium

[5] A. Cavallaro, O. Steiger, T. Ebrahimi, “Semantic Video Análisis for Adaptive Content Delivery and Automatic Description”, IEEE Transactions of Circuits and Systems for Video Technology, 15(10):1200-1209, Octubre 2005.

[6] A. Albiol, C. Sandoval, A. Albiol, “Robust Motion Detector for Video Surveillance Applications”. Proc. of the International Conference on Image Processing. Barcelona. Spain. 2003.

[7] Tadashi Nakanishi and Kenichiro Ishiim “Automatic vehicle image extraction based on spatio-temporal image analysis”, NTT Human Interface Laboratories 2003 Japan

[8] Farin, D. de With, P.H.N. Effelsberg, W., “Robust background estimation for complex video sequences”. Image Processing, 2003. ICIP 2003. Proceedings**.**

[9] U. Handmann, T. Kalinke, C. Tzomakas, M. Werner, and W. von Seelen, “Computer vision for driver assistance systems,” Proc. SPIE, Volume 3364, 1998, pp. 136–147.

[10] G. R. Bradski, “Computer vision face tracking as a component of a perceptual user interface,” in Proc.IEEE Workshop on Applications of Computer Vision, Princeton, NJ, October 1998, pp. 214–219.

[11] A. D. Bue, D. Comaniciu, V. Ramesh, and C. Regazzoni, “Smart cameras with real-time video object generation,” in Proc. IEEE Intl. Conf. on Image Processing, Rochester, NY, volume III, 2002, pp. 429–432.

[12] Rober J. Schalkoff. Digital Image Processing and Computer Vision. pp. 213-218. Editorial John Wiley & Sons, Inc, 1989.

[13] B.K.P. Horn. Robot Vision, MIT Press, Cambridge, MA, & McGrawn-Hill Book Company, New York, 1986.

[14] M. Wessler, L.A. Stein, “Robust Active Vision from Simple Symbiotic Subsystems”, Technical Report, MIT,1997.

[15] P. Anandan. “A computacional cuadrowork and an algorithm for the measurement of visual motion”, International Journal of Computer Vision, 2(3):283-310, January, 1989.

[16] S. A. Brock-Gunn, G.R. Dowling, T. J. Ellis. “Tracking using colour information”, Tech. Rep.TCU/CS/1994/7, City Univ. London, 1994.

[17] M. Kass, A. Witkin and Terzopoulos. “Snakes: Active contour models”, International Journal of Computer Vision, 1(4):133-144, 1987.

[18] Sylvia Gil, Ruggero Milanese, Thierry Pun. “Feature selection for object tracking in traffic scenes”, In SPIE International Symposium on Smart Highways, Boston, Massachusetts, Oct. 31- Nov. 4, 1994.

[19] E. Sánchez-Nielsen, M. Hernández-Tejera. "Tracking moving objects using the Hausdorff distance. A method and experiments" Frontiers in Artificial Intelligence and

Applications: Pattern Recognition and Applications, M.I. Torres and A. Sanfeliu (eds.), pp. 164-172, IOS Press, Amsterdam (2000).
[20] W.J. Ruckelidge. “Efficient Computation of the minimum Hausdorff Distance for Visual Recognition”, Phd thesis, Cornell Universitym 1995. CS-TR1454
[21] G. Medioni, “Detecting and tracking moving objects for video surveillance,” in Proceedings of IEEE Int’l Conference on Computer Vision and Pattern Recognition, 1999, vol. 2, pp. 319–325.
[22] B.P.L. Lo and S.A. Velastin, “Automatic congestion detection system for underground platforms,” Proc. of 2001 Int. Symp. on Intell. Multimedia, Video and Speech Processing, pp. 158-161, 2000.
[23] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati, “Detecting moving objects, ghosts and shadows in video streams”, IEEE Trans. on Patt. Anal. and Machine Intell., vol. 25, no. 10, Oct. 2003, pp. 1337-1342.
[24] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, “Pfinder:Real-time Tracking of the Human Body,” IEEE Trans. on Patt. Anal. and Machine Intell., vol. 19, no. 7, pp. 780-785, 1997.
[25] C. Stauffer, W.E.L. Grimson, “Adaptive background mixture modelsfor real-time tracking”, Proc. of CVPR 1999, pp. 246-252.
[26] Elgammal, A., Harwood, D., and Davis, L.S., “Non-parametric Model for Background Subtraction”, Proc. of ICCV '99 FRAME-RATE Workshop, 1999.
[27] B. Han, D. Comaniciu, and L. Davis, "Sequential kernel density approximation through mode propagation: applications to background modeling,“ Proc. ACCV -Asian Conf. on Computer Vision, 2004.
[28] N. M. Oliver, B. Rosario, and A. P. Pentland, “A Bayesian Computer Vision System for Modeling Human Interactions,” IEEE Trans. on Patt. Anal. and Machine Intell., vol. 22, no. 8, pp. 831-843, 2000.
[29] Haoran Yi, Deepu Rajan, Liang-Tien Chia. “Automatic Generation of MPEG-7 Compliant XML Document for Motion Trajectory Descriptor in Sports Video”. Multimedia Tools Appl. 26(2): 191-206 (2005)
[30] J.M. Martínez (ed.): *“MPEG-7 Overview (Version 10)”*, ISO/IEC JTC1/SC29/WG11N6828. 70th MPEG meeting Palma de Mallorca, Octubre 2004. Disponible en: http://www.chiariglione.org/mpeg/standards/MPEG7/MPEG7.htm
[31] B.S. Manjunath, Philippe Salembier, Thomas Sikora. “Introduction to MPEG-7. Multimedia Content Description Interface”,Wiley.
[32] “MPEG-7 Part 3”, ISO/IEC FDIS 15938-3:2002
[33] “MPEG-7 Part 5”, ISO/IEC FDIS 15938-5:2002
[34] “Extensible Markup Language (XML) 1.0 (Second Edition)”, W3C Recommendation 6 October 2000 http://www.w3.org/TR/REC-xml
[35] William H. Press, Saul A.Teukolsky, William T. Vetterling, Brian P. Flannery. “Numerical Recipes in C – The art of Scientific Computing 2nd Edition”. Cambridge University Press
[36] “Thinking in C++ 2nd Edition”, Bruce Eckel, http://mindview.net/Books/TICPP/ThinkingInCPP2e.html
[37] Richard C. Leinecker y Tom Archer .“Visual C++ 6.0” , Anaya Multimedia

Glosario

API	Application Programming Interface
CBR	Constant BitRate
CPU	Central Proccess Unit
CS	Classification Scheme
DDL	Description Definition Language
DS	Descriptor Scheme
DTD	Document Type Definition
FDIS	Final Draft International Standard
HTML	HyperText Markup Language
HTTP	HyperText Transport Protocol
IPMP	Intellectual Property Management and Protection
IS	International Standard
JNI	Java Native Interface
JVM	Java Virtual Machine
JWSDP	Java Web Services Development Pack
MCDI	Multimedia Content Description Interface (MPEG-7)
MDS	Multimedia Description Schemes
MPEG	Moving Picture Experts Group
PDA	Personal Digital Assistant
QoS	Quality of Service
SGML	Standard Generalized Markup Language
UMA	Universal Multimedia Access (Acceso Multimedia Universal)
URI	Uniform Resource Identifier
URL	Universal Resource Locator
TR	Technical Report
W3C	World Wide Web Consortium
XML	eXtensible Markup Language
XSL	eXtensible Stylesheet Language

Anexos

A Manual de instalación

Para la instalación del sistema completo, se necesitara el sistema operativo Windows XP instalado en el PC.

Para la aplicación transmisora solamente se necesitara el fichero ejecutable (Tx.exe) que contendrá todas las rutinas y el entorno visual que se ha mencionado en apartados anteriores.

Los datos de entrada de la aplicación transmisora deberán ser imágenes de la secuencia a analizar, dichas imágenes tendrán como nombre “imageXXX.bmp” y se localizaran en la carpeta “images” que colgará del mismo directorio que la aplicación transmisora.

Los datos de salida que genera la aplicación se localizaran en la carpeta “output/tx/” (dicha carpeta será creada automáticamente) y estos serán los datos que el módulo de transmisión tendrá que codificar y transmitir.

Figura A-1. Directorio raíz de la aplicación principal

Para la aplicación receptora, se necesitarán dos programas externos y dos librerías dinámicas para su correcto funcionamiento. Dichos programas son el transcodificador de imágenes “bmptojpg.exe” y la herramienta de creación de vídeo “ffmpeg.exe”. El programa “ffmpeg.exe” necesitará el uso de dos librerías dinámicas (“AVSredirect.dll” y “pthreadGC2.dll”). Dichos programas se tendrán que colocar en el mismo directorio que la aplicación receptora “Rx.exe”.

Los datos de entrada se encontraran en la carpeta “rx”, cuyo contenido será el mismo que la carpeta “tx” (de la aplicación transmisora). Dicha carpeta se encontrara en el mismo directorio que la aplicación receptora.

Los datos de salida serán videos (en función del nivel de reconstrucción solicitado) en formato MPEG-2, que se guardaran en el mismo directorio en el que se encuentre la aplicación receptora.

Figura A-2. Directorio raíz de la aplicación receptora

Estos programas se podrán encontrar en la carpeta “software adicional”incluida en directorio principal del CD

B Codificación BMP

El formato BMP[14] (Windows BitMaP) es probablemente el formato de fichero para imágenes más simple que existe. Aunque teóricamente permite compresión, en la práctica nunca se usa, y consiste simplemente en una cabecera y a continuación los valores de cada píxel, comenzando por la línea de más abajo y terminando por la superior, píxel a píxel de izquierda a derecha.

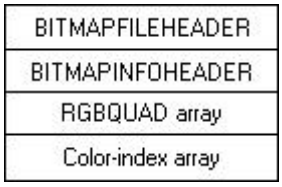
El formato BMP es el formato de las imágenes en bitmap de Windows. Aunque muy extendido, tiene la dificultad de la escasa compresión que realiza en los archivos por lo que ocupan rápidamente casi 1Mb. Pero el formato de Mapa de Bits tiene una importante característica a su favor, es que casi todos los usuarios tienen un PC que puede soportarlo.

Los archivos de mapas de bits se componen de direcciones asociadas a códigos de color, uno para cada cuadro en una matriz de píxeles tal como se esquematizaría un dibujo de "colorea los cuadros" para niños pequeños. Normalmente, se caracterizan por ser muy poco eficientes en su uso de espacio en disco, pero pueden mostrar un buen nivel de calidad. A diferencia de los gráficos vectoriales, al ser reescalados a un tamaño mayor, pierden calidad. Otra desventaja de los archivos BMP es que no son utilizables en páginas web debido a su gran tamaño en relación a su resolución.

Dependiendo de la profundidad de color que tenga la imagen cada píxel puede ocupar 1 o varios bytes. Generalmente se suelen transformar en otros formatos, como JPEG (fotografías), GIF o PNG (dibujos y esquemas), los cuales utilizan otros algoritmos para conseguir una mayor compresión (menor tamaño del archivo).

Los archivos comienzan (cabecera o header) con las letras 'BM' (0x42 0x4D), que lo identifica con el programa de visualización o edición. En la cabecera también se indica el tamaño de la imagen y con cuántos bytes se representa el color de cada píxel.

El formato de imagen BitMaP es el siguiente:



donde los miembros de la estructura *BITMAPFILEHEADER* identifican el archivo, el tamaño del mismo (en bytes) y el offset desde el primer byte de la cabecera al primer byte de los datos del mapa de bits.

[14] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/gdi/bitmaps_4v1h.asp

La estructura *BITMAPINFOHEADER* especifica la altura y la anchura del mapa de bits (en píxeles), el formato del color (número de planos de color y bits de color por píxel) del display donde fue creado el bitmap, la resolución de dicho display y el número de colores representados en los datos.

La estructura *RGBQUAD array* contiene la intensidad correspondiente a cada color que contiene el bitmap. Dicha intensidad esta especificada como una estructura de 3 cantidades que se corresponden con tres primarios: rojo, verde y azul (RGB).

Finalmente Color-index array asocia un color, contenido en RGBQUAD, con cada píxel en el bitmap. El número de bits de dicha estructura será igual al número de píxeles por el número de bits por píxel que usamos para representar el color de cada píxel.

C Dominant color

C.1 Introducción

Este descriptor especifica un conjunto de colores dominantes de una region con una forma arbitraria. El descriptor *Dominant Color* sirve tanto para una imagen entera como para una region arbitraria (rectangular o irregular).

C.2 Sintáxis DDL

```
<complexType name="DominantColorType" final="#all">
   <complexContent>
      <extension base="mpeg7:VisualDType">
         <sequence>
            <element name="ColorSpace" type="mpeg7:ColorSpaceType"
                  minOccurs="0"/>
            <element name="ColorQuantization"
                  type="mpeg7:ColorQuantizationType" minOccurs="0"/>
            <element name="SpatialCoherency" type="mpeg7:unsigned5"/>
            <element name="Value" minOccurs="1" maxOccurs="8">
               <complexType>
                  <sequence>
                     <element name="Percentage" type="mpeg7:unsigned5"/>
                     <element name="Index">
                        <simpleType>
                           <restriction>
                              <simpleType>
                                 <list itemType="mpeg7:unsigned12"/>
                              </simpleType>
                              <length value="3"/>
                           </restriction>
                        </simpleType>
                     </element>
                     <element name="ColorVariance" minOccurs="0">
                        <simpleType>
                           <restriction>
                              <simpleType>
                                 <list itemType="mpeg7:unsigned1"/>
                              </simpleType>
                              <length value="3"/>
                           </restriction>
                        </simpleType>
                     </element>
                  </sequence>
               </complexType>
            </element>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

Figura C.1. Esquema DDL del descriptor Dominant Color

C.3 Representación binaria del esquema

DominantColor {	**Number of bits**	**Mnemonic**
Size	3	uimsbf
ColorSpacePresent	1	bslbf
if(ColorSpacePresent) {		
ColorSpace	See subclause	ColorSpaceType
}		
ColorQuantizationPresent	1	bslbf
if(ColorQuantizationPresent) {		
ColorQuantization	See subclause	ColorQuantizationType
}		
VariancePresent	1	bslbf
SpatialCoherency	5	uimsbf
for(k=0; k<Size; k++) {		
Percentage	5	uimsbf
for(m=0; m<3; m++) {		
Index	1-12	uimsbf
if(VariancePresent) {		
ColorVariance	1	uimsbf
}		
}		
}		
}		

Tabla C.2. Representación binaria del Esquema DDL del descriptor Dominant Color

C.4 Componentes del descriptor

Size

Este campo, que está solamente presente en la representación binaria, especifica el número de colores dominantes en la región. El número permitido máximo de colores dominantes es 8, el número mínimo de colores dominantes es 1. El traz siguiente de configuraciones de bits se utiliza: 000->1,…, 111->8.

ColorSpacePresent

Este campo, que está solamente presente en la representación binaria, indica la presencia del elemento de ColorSpace. Si su valor es 0, ColorSpace no está presente y se utiliza el espacio de color del RGB.

ColorSpace

Este elemento se define en subclaúsula.

ColorQuantizationPresent

Este elemento, que está solamente presente en la representación binaria, señala la presencia del elemento de ColorQuantization. Si su valor es 0, ColorQuantization no está presente y el quantization uniforme del color de los componentes a 5 pedacitos se utiliza.

ColorQuantization

Este elemento se define en subclaúsula.

VariancePresent

Este campo, que está solamente presente en la representación binaria, indica la presencia de las variaciones del color en el descriptor.

SpatialCoherency

Este elemento especifica la coherencia espacial de los colores dominantes descritos por el descriptor. Es computado como un solo valor por la suma cargada de coherencias espaciales del por-dominante-color. El peso es proporcional al número de los pixeles que corresponden a cada color dominante. La coherencia espacial por capturas dominantes del color cómo es coherente son los pixeles que corresponden al color dominante y si aparecen ser un color sólido en la región dada de la imagen (véase el cuadro 1, donde los pixeles rojos en la imagen izquierda tienen coherencia espacial baja y en la alta coherencia espacial de la imagen derecha). La coherencia espacial por color dominante es computada por la conectividad media normalizada (8-connectedness) para los pixeles dominantes correspondientes del color.

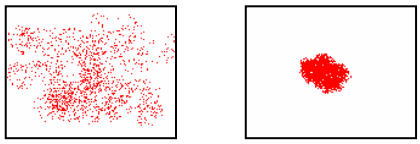


Figure 1 — Ejemplos de (a) baja y alta (b) coherencia espacial del color

La suma cargada de coherencias espaciales del por-dominante-color se normaliza a partir de la 0 a 1, entonces no-uniformemente cuantificado a la gama a partir de la 1 a 31 como sigue. Los valores normalizados menos de 0.7 se fijan a 1, mientras que los valores entre 0.7 a 1 son uniformemente cuantificado a la gama 2 a 31. 0 se utilizan señalar que este elemento no está computado (la nota que si no se computa él no significa que la coherencia espacial es baja).

Value

Este elemento especifica un arsenal de los elementos que llevan a cabo porcentajes y valores de colores en un artículo visual. Los elementos de arsenal consisten en porcentaje, ColorValueIndex y ColorVariance.

Percentage

Este elemento especifica el porcentaje de los pixeles que tienen el valor asociado del color. El valor del porcentaje es uniformemente cuantificado a 5 bits con 0 que corresponde a 0 porcentajes y 31 que corresponden a 100%. Observar que la suma de los valores del porcentaje para un ítem visual dado no tiene que ser igual a 100%.

Index

Este elemento especifica el índice del color dominante en el espacio de color seleccionado según lo definido en ColorQuantization. El número de los bitspedacitos para cada componente se deriva del elemento de ColorQuantization.

ColorVariance

Este elemento especifica un arsenal del número entero que contiene el valor de la variación de los valores del color de los pixeles que corresponden al color dominante en el espacio de color seleccionado, es decir.

$$CV_j = \frac{1}{N} \sum_{k=0}^{N-1} \left(m_j - p_{kj} \right)^2$$

donde j pone indexa el componente del color, el mj es componente del j-th del color dominante, el pkj es componente del j-th del valor del píxel del k-th, y la adición es píxeles excesivos de N que corresponden al color dominante bajo consideración. La dimensión de este vector depende del espacio de color seleccionado. Cada componente es cuantificado a 1 bit, con “0” que corresponde a la variación baja y “1” correspondiendo a la alta variación. El umbral del cuantificación es igual a 0.005 de la gama componente ajustada del valor del color.

D MPEG-7: Estándar de descripción de contenido multimedia

D.1 Introducción

Hace años el acceso a contenidos multimedia solía ser una tarea que no entrañaba ningún problema. Esto era debido a la manera de acceder a ellos y que existía disponible poco contenido multimedia. No existía una necesidad de clasificar el contenido multimedia

Actualmente una enorme y creciente cantidad de contenido audiovisual se ha puesto disponible a todo el mundo gracias a la revolución digital y se ha difundido principalmente a través de internet. En esta nueva situación, el valor de la información empieza a depender de cómo es el acceso a ella. Es decir tenemos un acceso complejo debido a la gran cantidad de información y fuentes que la distribuyen. Así pues el valor de la información ya no dependerá solamente de ella misma, sino también de la facilidad para ser encontrada, recuperada y filtrada.

En la transición del segundo al tercer milenio han surgido nuevas formas de producir, filtrar, buscar y administrar información multimedia digitalizada. Soluciones de banda ancha se vuelven cada vez más accesibles a los usuarios, ofreciendo crecientes velocidades de acceso y calidades de audio y vídeo. La tendencia está clara, en los próximos años los usuarios se enfrentarán a una tal cantidad de contenidos disponibles a través de una gran cantidad de fuentes que el acceso eficiente y preciso se vuelve algo muy valioso. A pesar de que los usuarios cada vez tienen mayor acceso a estos contenidos, identificar y administrarlos eficientemente se está volviendo cada vez más complicado, tanto a nivel profesional como personal.

MPEG-7 trata de ser la solución a esta necesidad. MPEG-7 es un estándar internacional de la ISO/IEC, desarrollado por el grupo MPEG (Moving Picture Experts Group), que ha desarrollado otros estándares tan importantes como son el MPEG-1, MPEG-2 y MPEG-4. MPEG-1 y MPEG-2 han permitido la producción de productos ampliamente conocidos y adoptados, como el Video CD, MP3, DVD, DAB, la televisión digital entre otros. MPEG-4 es el primer estándar real de representación multimedia, permitiendo interactividad y la combinación de material natural y sintético, codificado como objetos.

El estándar MPEG-7, conocido como *Interfaz de Descripción de Contenido Multimedia*, proporciona un amplio conjunto de herramientas estándar para describir contenido multimedia, tanto para usuarios humanos como para sistemas automáticos que procesen información audiovisual. Estas herramientas de descripción (conocidas como Description Tools, que son los elementos de metadatos y su estructura y relaciones, definidas en el estándar como Descriptores (D) y Esquemas de Descripción (DS)) sirven para crear descripciones que serán la base para

aplicaciones que permitan este necesario acceso eficiente a contenido multimedia. Es una tarea complicada dado el amplio espectro de requisitos y aplicaciones multimedia que pretende abarcar, así como el gran número de características audiovisuales de importancia en este contexto.

D.2 Contexto

El contenido multimedia juega un papel importante en nuestra sociedad. Cada dia mas y mas cantidad de información esta disponible desde un número creciente de fuentes. Dichas fuentes pueden ser de pueden ser de diversos tipos: imágenes, gráficos, modelos 3D, audio, habla, vídeo...

La información audiovisual generalmente es consumida directamente por el ser humano, aunque hay un creciente número de casos en los que la información audiovisual es creada, intercambiada, recuperada y reusada por sistemas computacionales, como puede ser el caso de la comprensión y reconocimiento de imágenes (vigilancia, visión artificial, cámaras inteligentes, etc.) y la conversión de medio (de voz a texto, de imagen a voz, etc.).

La necesidad de un procesado posterior de un contenido audiovisual hace necesario el desarrollo de nuevas formas de representación que vayan más allá de la simple representación basada en forma de onda (waveform), o comprimida, o incluso basada en objetos (MPEG-4), de forma que se permita un cierto grado de interpretación del significado de la información si es necesario.

MPEG-7 es un estándar de descripción de contenido multimedia que soportará estos requisitos operacionales. Estos requisitos se aplican tanto para aplicaciones de tiempo real y de no tiempo real. MPEG-7 no estandariza ni evalúa aplicaciones.

D.3 Objetivos

Desde que *MPEG* empezase a trabajar en *1996* con el futuro estándar, un nuevo camino se ha abierto para solucionar los problemas antes descritos. No se trata de un estándar a la manera “tradicional” de *MPEG*, ya que no se representa el contenido, sino que se va a representar información sobre ese contenido (“los *bits* sobre los *bits*”).

Las herramientas de descripción que ofrece *MPEG-7* son, por tanto, independientes de cómo está almacenado o codificado el contenido (se puede trabajar con señales analógicas o digitales). Parte de la base de que existe una representación (estandarizada o no) del contenido y se encarga de describir todo lo relacionado con él.

Debido a que las posibilidades que existen para describir información son casi infinitas, el estándar debe barrer diferentes niveles de abstracción. Éstos van desde características de bajo nivel (en relación directa con el vídeo o el audio, por ejemplo) que pueden ser extraíbles de forma

automática, hasta altos niveles (descripciones de tipo semántico) donde se necesitan personas para su extracción.

Se observa que la información textual para los descriptores es muy útil para el caso de descripciones de tipo semántico, pero el estándar no quiere estancarse en este único modo, y propone hacer uso de todo tipo de información (histogramas, fragmentos de audio, etc.) para trabajar con el contenido.

MPEG-7 no busca encasillarse en una parcela determinada de la información multimedia o de aplicación, como ocurre con otros estándares tales como *SMPTE Metadata Dictionary*[15], *EBU P/Meta*[16], *Dublin Core*[17] y *TV Anytime*[18] entre otros, su rango de actuación incluye multitud de aplicaciones en entornos diferentes, lo que conlleva la necesidad de ofrecer un marco de trabajo flexible y variable para describir datos multimedia. Hay que señalar, por tanto, que *MPEG-7* no define un sistema monolítico para describir el contenido, sino un conjunto de métodos y herramientas para las diferentes posibilidades de descripción de un mismo contenido. *MPEG-7* intenta ser lo más genérico posible.

MPEG-7 utiliza *XML Schema* como lenguaje para la representación textual del contenido, lo cuál le permite ser flexible y aumentar las herramientas de descripción existentes. Para poder dar al estándar mayores funcionalidades, no se utiliza *XML Schema* puro, sino que se han añadido algunas extensiones de las que éste carece.

Los elementos principales del estándar *MPEG-7* son:

- ***Descriptores* (*D*)**: son las representaciones de características que definen la sintaxis y la semántica de cada representación característica.
- ***Esquemas de Descripción* (*DS*)**: especifican la estructura y semántica de las relaciones entre sus componentes. Estos componentes pueden ser tanto *D's* como *DS's*.
- ***Lenguaje de Definición de Descripciones* (*DDL*)**: proporciona las herramientas para permitir la creación de nuevos *DS's* y *D's*. También permite la extensión y modificación de los *DS's* existentes.

---

[15] http://www.smpte-ra.org

[16] http://www.ebu.ch/ pmc_metal.html

[17] http://dublincore.org

[18] http://www.tv-anytime.org

- ***Herramientas del sistema***: permiten el multiplexado de descripciones y la sincronización de las mismas con el contenido. Se ocupan de los mecanismos de transmisión, representaciones codificadas (tanto textuales como binarias) para almacenarlas y transmitirlas de forma eficiente. También incluyen los procesos de gestión y protección de la propiedad intelectual, etc. cuya relación se puede observar en la Figura *2.3*.

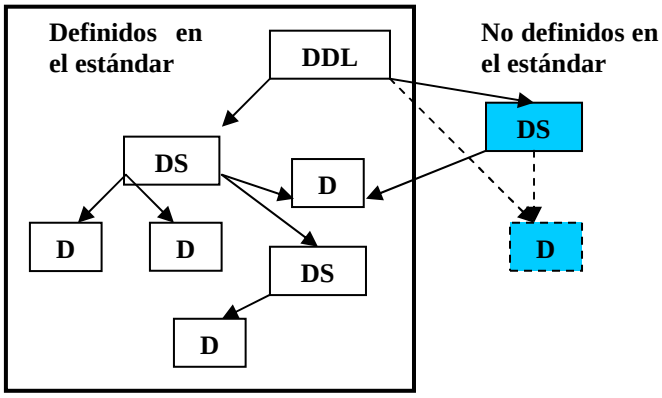


Figura 2-2 Relación entre elementos de MPEG-7

Estos *DS's* y *D's* describen el contenido desde distintos puntos de vista definidos por *MPEG-7*:

- ***Creación y Producción***: elementos relacionados con la creación y la producción del contenido, tales como: título, creador, clasificación, propósito de la creación, etc. La mayoría de esta información no se puede extraer directamente del contenido.
- ***Uso***: elementos que describen la información acerca del uso del contenido, como pueden ser: derechos de acceso, información financiera, etc. Este tipo de información está sujeta a cambios a lo largo de la vida del contenido audiovisual.
- ***Medio***: estos elementos describen el almacenamiento, por ejemplo: formato de almacenamiento o codificación del contenido audiovisual entre otros.
- ***Aspectos Estructurales***: contiene los elementos de descripción de la estructura en torno a segmentos que representan componentes físicos espaciales, temporales o espacio-temporales del contenido audiovisual. Cada uno de estos segmentos puede estar descrito por características básicas como el color, la textura o algún otro tipo de información semántica.
- ***Aspectos Conceptuales***: describe el contenido audiovisual a través de sus nociones conceptuales.

- ***Otros Aspectos***: aparte de estos aspectos, directamente relacionados con la descripción del contenidos, *MPEG-7* estandariza herramientas de descripción para dar soporte a ciertas aplicaciones. Estos aspectos son Navegación y Acceso, Organización del Contenido, e Interacción del usuario.

Los tres primeros conjuntos están relacionados con la información desde el punto de vista de la gestión de contenido, mientras que los dos últimos se refieren a la descripción de contenido.

D.4 Alcance

La siguiente figura muestra un diagrama de bloques de la posible cadena de procesado en MPEG-7, donde se aprecia cual es el alcance del estándar:

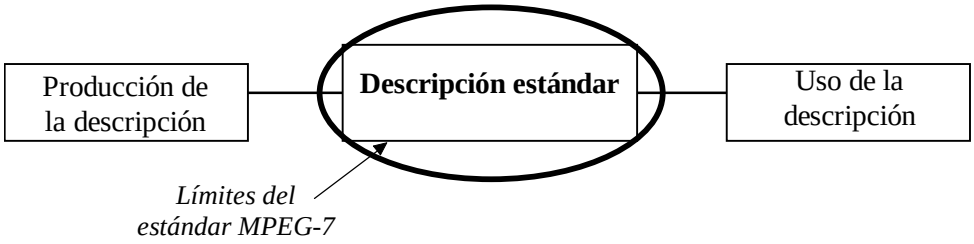


Figura 0-3: Alcance de MPEG-7

En primer lugar tenemos la extracción de características (análisis), que puede ser o no automática. De ahí pasa a la descripción en sí misma, para acabar en el motor de búsqueda (aplicación). Aunque la extracción automática o semiautomática de características es muy útil, queda fuera del estándar, ya que no es necesario para permitir la interoperabilidad, dejando así que sea la industria quien proponga soluciones. Igualmente ocurre con los motores de búsqueda, agentes de filtrado o cualquier aplicación que haga uso de las descripciones, no se estandarizan y se deja que sea la industria quien promueva la competición de soluciones.

El estándar desea que se presenten soluciones para la extracción y la búsqueda sin querer adoptar ninguna de ellas. Se limita a estandarizar la forma en que se codifica la descripción del contenido. El uso, o la manera de generarlo queda abierto a cualquiera que desee ofrecer una solución, potenciando así la competición entre diferentes soluciones. Un ejemplo de cómo puede quedar la cadena de todo el proceso se representa en la siguiente figura, donde las formas rectangulares representan herramientas funcionales, como codificación o descodificación, mientras que las circulares representan elementos estáticos, por ejemplo, descripciones. Los rectángulos de fondo gris son los elementos normativos que se acaban de definir. En esta figura también se incluyen elementos que no están incluidos en el estándar como puede ser el proceso de extracción o el sistema de búsqueda, aunque sí estén incluidos en el software de referencia.

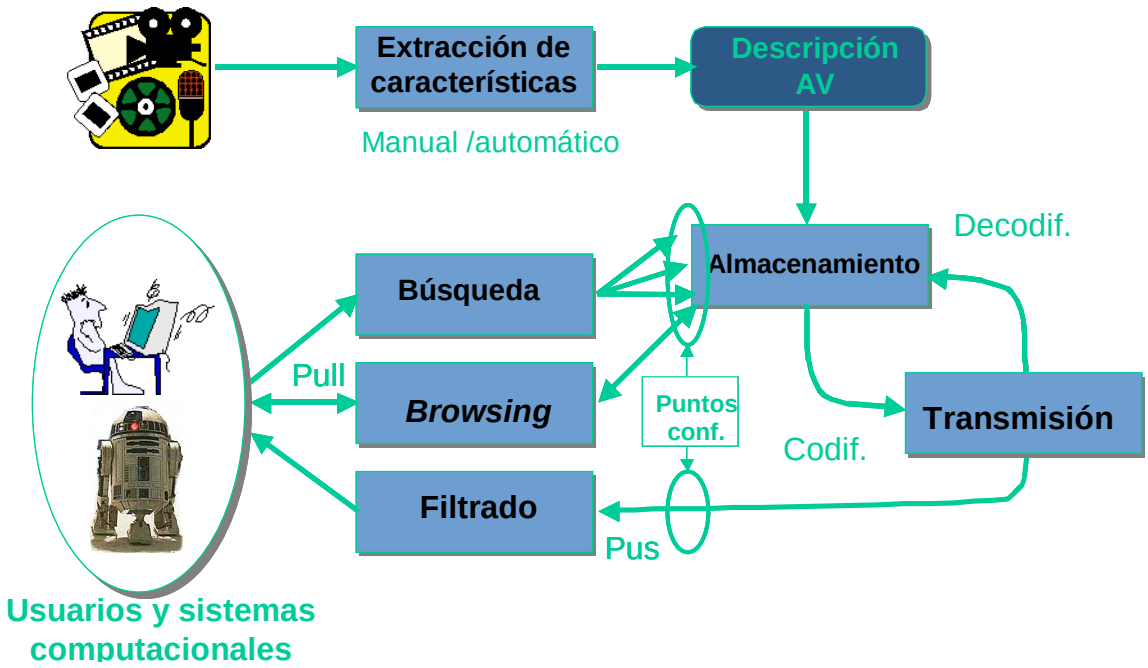


Figura 0-4: Representación de una posible aplicación MPEG-7

D.5 Áreas de aplicación

Como ya se ha mencionado, los elementos que MPEG-7 va a estandarizar van a poder utilizarse en un amplio rango de aplicaciones. Todos los dominios de aplicaciones que hagan uso de contenido multimedia se podrán beneficiar de este nuevo estándar. Algunas posibles aplicaciones son, por ejemplo, las siguientes:

- Arquitectura, inmobiliarias (compra-venta de pisos) y diseño de interiores (p. ej.: búsqueda de ideas).
- Comercio electrónico (p. ej.: anuncios personalizados, catálogos en línea, directorios de tiendas electrónicas).
- Compras (búsqueda de vestidos, etc.).
- Control remoto (p. ej.: cartografía, ecología, gestión de recursos naturales).
- Edición multimedia (servicios de noticias electrónicas personalizado, etc.).
- Educación (almacén de cursos multimedia, búsqueda multimedia para material de ayuda, etc.).
- Entretenimiento (p. ej.: sistemas para gestión de colecciones multimedia).
- Librerías digitales (p. ej.: catálogos de imágenes, diccionarios musicales, archivos de películas y radio).
- Periodismo (búsqueda de discurso de una persona usando su nombre, su voz, su cara, etc.).
- Selección de canales de difusión (p. ej.: canales de radio, de televisión).

- Servicios de cultura (museos de historia, galerías de arte, etc.).
- Servicios de directorio multimedia (p. ej.: páginas amarillas, información a turistas, sistemas de información geográfica).
- Servicios de investigación (p. ej.: reconocimiento de características humanas, medicina forense).
- Servicios sociales (p. ej.: servicio de citas).
- Vigilancia (control del tráfico, transporte de superficie, pruebas no destructivas en lugares hostiles).

D.6 Partes de MPEG-7

El estándar ISO/IEC 15938 está formado por las siguientes partes que ya son estándar internacional (IS), siguiendo la actividad en alguna de ellas:

- MPEG-7 Systems: arquitectura del estándar y herramientas necesarias para preparar las descripciones de MPEG-7 para su transporte y almacenamiento eficiente. También se ocupa de conseguir la sincronización entre contenido y descripciones. Otro aspecto que engloba son las herramientas relacionadas con la gestión y protección de la propiedad intelectual.

- MPEG-7 Description Definition Language: lenguaje para definir nuevos DSs y, quizás eventualmente, también para nuevos Ds. Está basado en el XML Schema Language (XSL), aunque necesita de ciertas extensiones específicas para MPEG-7, ya que no fue diseñado para describir material audiovisual.

- MPEG-7 Audio: estructuras básicas y descriptores que cubren las siguientes características visuales básicas: color, textura, forma, movimiento, localización y reconocimiento facial. Cada categoría cuenta con un descriptor elemental y específico.

- MPEG-7 Visual: proporciona estructuras para describir material sonoro. En ellas se tiene un conjunto para de descriptores de bajo nivel (para características espectrales, temporales, etc), y otro de herramientas de alto nivel (que incluye herramientas para el reconocimiento de sonidos e indexación, para descripción timbral de instrumentos, para habla, y para audio)

- MPEG-7 Multimedia Description Schemes: elementos (DSs y Ds) que describen información genérica, es decir, que no es ni puramente visual ni de audio.

- MPEG-7 Reference Software - the eXperimentation Model: es una plataforma de simulación para los descriptores (Ds), Esquemas de descripción (Description Schemes

or DSs), Esquemas de codificación (Coding Schemes or CSs) y el lenguaje de definición de descripciones (Data Definition Language or DDL). Ademas de los componentes normativos, la plataforma de simulación necesita tambien otro componentes que no son normativos, esencialmente para ejecutar algunos procedimientos sobre las estructuras de datos. Las estructuras de datos y el codigo de los procedimientos forman las aplicaciones. Las aplicaciones XM están divididas en dos tipos: las aplicaciones servidoras (extracción) y las aplicaciones cliente (búsqueda, filtrado y/o transcodificación).

- MPEG-7 Conformance Testing: guía y procedimientos para comprobar que las implementaciones de MPEG-7 se ajustan a las normas establecidas (en desarrollo, en fase FDIS).

- MPEG-7 Extraction and use of the descriptions: material informativo (a modo de informe técnico) sobre la extracción y uso de algunas de las herramientas de descripción.

- MPEG-7 Profiles: recoge distintos perfiles y niveles estandarizados para MPEG-7, especificados a través de las partes del estándar ISO/IEC 15938. Mientras todas las partes son potencialmente candidatas para perfiles, los perfiles actuales se concentran en las partes Description Definition Language [ISO/IEC 15938-2], Visual [ISO/IEC 15938-3], Audio [ISO/IEC 15938-4], Multimedia Description Schemes [ISO/IEC 15938-5], las cuales están basadas en Schema Definition [ISO/IEC 15938-10].

- MPEG-7 Schema Definition: recoge todos los esquemas MPEG-7, reuniéndolos desde diferentes estándares

- MPEG-7 Profiles Schema Definition: recoge los esquemas de los perfiles MPEG_7.

Además de los documentos publicados, existen otros que son correcciones (Amendments) de algunas de las partes de MPEG-7, y que se encuentran en diferentes fases de desarrollo:

- MPEG-7 Part 1: Systems AMENDMENT 1: System Extensions
- MPEG-7 Part 3: Visual AMENDMENT 1: Visual Extensions
- MPEG-7 Part 4: Audio AMENDMENT 1: Audio Extensions
- MPEG-7 Part 5: Multimedia Description Schemes AMENDMENT 1: Multimedia Description Schemes Extensions
- MPEG-7 Part 6: Reference Software AMENDMENT 1: Reference Software Extensions

- MPEG-7 Part 7: Conformance Testing AMENDMENT 1: Conformance Testing Software Extensions

D.7 Esquemas de Descripción Multimedia (MDS) de MPEG-7

Tal y como se ha mencionado anteriormente, uno de los elementos principales del estándar son los DS's, los cuales están constituidos por D's u otros DS's. En este apartado se encuentran los DS's que no son propiamente de audio ni de vídeo, sino genéricos.

Los D's de MPEG-7 se utilizan para describir los siguientes tipos de información:

- Características audiovisuales de bajo nivel, como son el color, la textura, el movimiento, la energía, etc.
- Características de alto nivel de objetos semánticos, eventos y conceptos abstractos.
- Procesos de gestión del contenido.
- Información sobre el contenido.

Se supone que gran parte de los D's que describen características de bajo nivel podrán ser extraídos automáticamente, mientras que se necesitará más intervención humana para producir D's de alto nivel.

Los DS's definidos en MPEG-7 expanden los D's del estándar al combinar tanto D's individuales como otros DS's en estructuras más complejas. A su vez, se definen las relaciones entre los DS's y sus D's constituyentes.

En MPEG-7, los DS's están clasificados en tres grandes grupos: los que pertenecen a audio, los que lo hacen al dominio visual, o los que se agrupan en torno a la descripción genérica de datos multimedia. Por ejemplo, este último grupo describe los metadatos relacionados con la creación, producción, uso y gestión de los datos multimedia; así como el contenido multimedia a distintos niveles de abstracción (incluyendo estructura de la señal, características modelos y semántica). Este grupo describe todo tipo de información multimedia (datos de tipo visual, textual o sonoro), mientras que los D's específicos, como son color, textura, forma, melodía, etc. pertenecen, según sea su naturaleza, a los DS's del dominio visual o de audio. Como ocurría con los D's que sirven para describir características de bajo nivel, la instanciación de los DS's puede hacerse en algunas ocasiones de forma automática, pero la mayoría de las veces será manual.

Se muestra un resumen de la organización de los Esquemas de Descripción Multimedia, que se dividen en:

- Elementos básicos y herramientas de esquema.

- Descripción del contenido.
- Gestión del contenido.
- Organización del contenido.
- Navegación y acceso.
- Interacción con el usuario.

En la siguiente figura se muestran gráficamente esta organización. Los cuadros sombreados se corresponden con los grandes grupos de DS's de los que hace uso el sistema de este proyecto. En un futuro se podrá extender a otros grupos de descriptores para aumentar la funcionalidad del sistema.

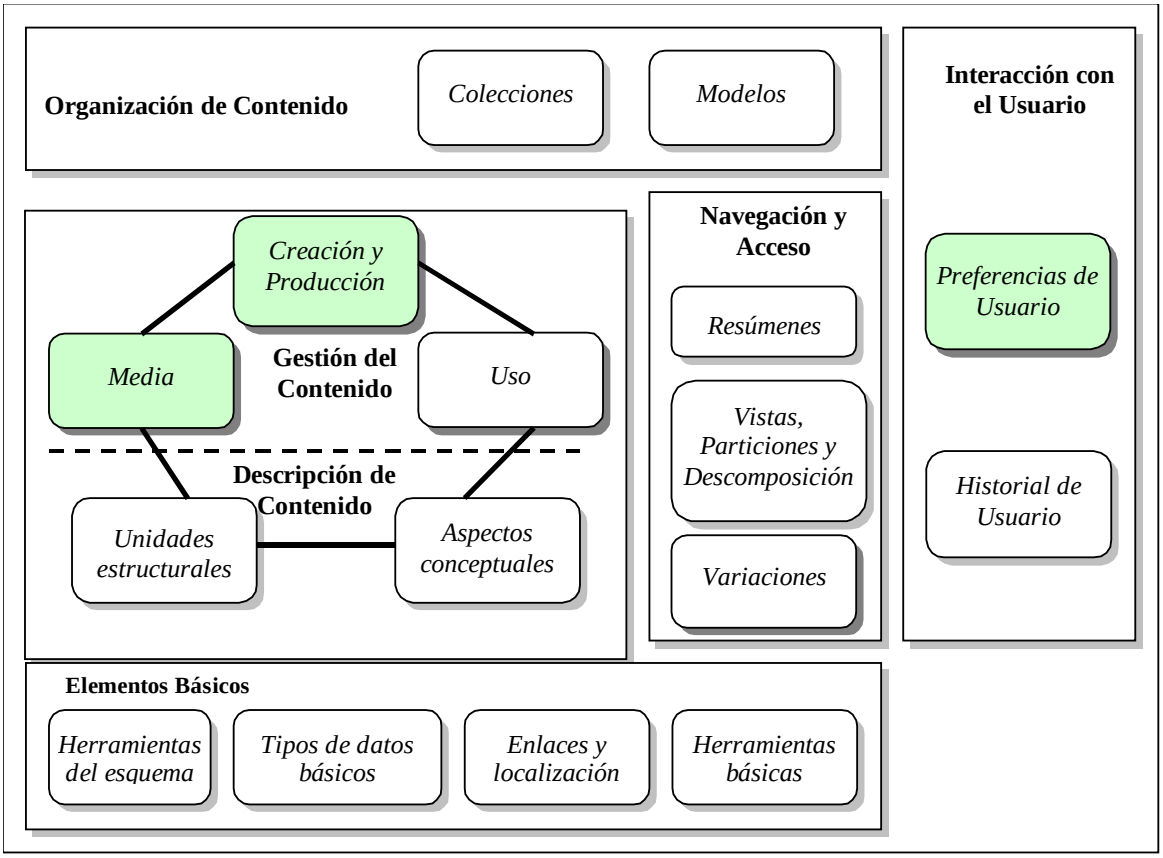


Figura 0-5: Resumen de los Esquemas de Descripción Multimedia

D.7.1 Elementos Básicos

La especificación de los MDS's define un gran número de elementos básicos que se usan como constructores fundamentales en la definición de muchos de los DS's de MPEG-7. Muchos de los elementos básicos ofrecen tipos de datos específicos y estructuras matemáticas, tales como vectores, matrices e histogramas, los cuales son de gran utilidad para la descripción del contenido multimedia. También se consideran como elementos básicos aquellos que relacionan ficheros de

datos diferentes y localizan segmentos, regiones, etc. Estos elementos cubren la mayoría de las necesidades que aparecen en la descripción de contenidos multimedia, tales como el tiempo, lugares, personas, grupos, organizaciones y varios tipos de anotaciones de tipo textual.

Se pueden distinguir dos grandes grupos en estos elementos básicos, los cuales son utilizados por la mayoría de DS's de MPEG-7:

- Información temporal. Los DS's que describen el tiempo se basan en el estándar ISO 8601, el cual también ha sido adoptado por el lenguaje XML *Schema* (usado como base para el DDL). Los DS's de tiempo representan la información temporal del contenido ("*streams*" de los datos) o de lo que éste representa. MPEG-7 extiende esta especificación (ISO 8601) con el objeto de describir también el tiempo a nivel de muestras de la codificación del contenido multimedia, como puede ser cierta cantidad de periodos del muestreo digital.

- Anotación textual. Los DS's y D's que sirven para hacer anotaciones textuales también son componentes importantes de otros muchos DS's. MPEG-7 aporta una serie de construcciones básicas para ésta, incluyendo texto libre (palabras clave, frases), texto estructurado (texto y reglas), y anotación estructurada relacional (texto estructurado y relaciones). Con este bloque se pretende abarcar un amplio rango de funcionalidades de descripción textual.

La especificación del MDS también ha definido un conjunto de Herramientas de Esquema que facilitan la creación y agrupación de descripciones MPEG-7. Estas herramientas consisten en un conjunto de elementos y de herramientas: elementos base (*base*), elementos raíz (*root*), elementos de nivel superior (*top-level*) y herramientas de empaquetado (*packages*). Los elementos base son elementos abstractos que definen la jerarquía de las descripciones. Los elementos raíz, sobre los que se definen en las descripciones MPEG-7, permiten la creación de fragmentos o documentos XML según MPEG-7. Los elementos de nivel superior, inmediatamente inferiores en la jerarquía a los elementos raíz, organizan los DS's con los que se está trabajando en ese momento para tareas completamente específicas orientadas a la descripción del contenido, como pueden ser:

- Describir una imagen, vídeo, audio o contenido multimedia.
- Crear colecciones.
- Describir preferencias usuarios.
- Mostrar la semántica.

Por último, las herramientas de agrupación se utilizan para asociar ciertos componentes de los DS's a determinadas carpetas o grupos. Estos grupos son útiles para organizar y trasladar la

estructura y los tipos de información de descripciones MPEG-7 a motores de búsqueda ya existentes con GUI's (*Graphic User Interfaces*) propietarios no basados en un modelo de metadatos MPEG-7. Actúan como traductores/adaptadores de metadatos de diversos modelos.

D.7.2 Gestión del Contenido

MPEG-7 ofrece DS's para la descripción de los diferentes aspectos de la creación, producción, codificación, almacenamiento, formatos de archivos y uso del contenido. La funcionalidad de cada uno de los grupos de esta parte del estándar es:

- *Creation Information*: se encarga de describir la creación y clasificación del contenido multimedia y otros materiales que estén relacionados con ese contenido. Tiene tres grandes componentes: el primero se llama *Creation*, y ofrece herramientas de descripción para el título (puede ser textual o algún tipo de contenido multimedia) y la información sobre la creación del contenido (autores, lugares y fechas de la creación). El siguiente es *Classification*, que describe como se clasifica el material multimedia en categorías tales como el género, tema, propósito, lenguaje, etc. También ofrece herramientas para describir aspectos como la clasificación por edades. Finalmente, está *Related Material*, que describe si existe algún otro tipo de material multimedia relacionado con el contenido descrito.

- *Usage Information*: describe la información sobre el uso que se le puede dar al contenido multimedia (derechos de uso, disponibilidad e información financiera). La información sobre los derechos no está explícitamente incluida en la descripción MPEG-7, sin embargo, hay enlaces a los dueños de los derechos y a más información relacionada con la gestión y protección de los mismos. Existen cuatro grandes componentes en este grupo. *Rights*, que da referencias a los propietarios de los derechos o información relacionada, bajo la forma de identificadores unívocos gestionados por autoridades externas. Con ello se pretende que las descripciones den acceso a información sobre el dueño de los derechos sin tener que entrar en negociaciones previas. *Availability* y *Usage Record* ofrecen información relacionada con la disponibilidad y el uso pasado del contenido (difusión, demanda de entrega, etc.). Por último, *Financial* describe la información sobre el coste de la producción y los ingresos resultantes de su uso. Todos estos datos suelen variar con el tiempo y, por tanto, están sujetos al cambio durante la existencia del contenido multimedia.

- *Media Information*: describe información sobre el almacenamiento del contenido, como puede ser la compresión, codificación y formato de los datos. El componente

Media Identification identifica una entidad que tiene al menos una copia original (*master*), que es la fuente de donde se producen diferentes instancias19 del contenido audiovisual. Las instancias son versiones del original o de otros perfiles (las copias digitales son perfectas) a diferentes formatos de codificación, almacenamiento o entrega. *Media Profile* describe individualmente cada una de las copias por medio de sus parámetros de codificación, información y localización del almacenamiento.

D.7.3 Descripción del Contenido

Estos DS's describen la estructura (regiones, cuadros de vídeo y segmentos de audio) y la semántica (objetos, eventos y nociones abstractas) del contenido multimedia.

La funcionalidad de estas dos clases es:

- Aspectos estructurales: describen el contenido multimedia desde el punto de vista de la estructura. Estos DS's se organizan en torno a *Segment*, que representa la estructura espacial, temporal o espacio-temporal del contenido multimedia. *Segment* puede organizarse dentro de una estructura jerárquica para producir una tabla de contenidos que permita el acceso o indexado del contenido al hacer búsquedas. Los segmentos se describen utilizando DS's (anotaciones textuales) y D's de la parte de audio y vídeo.

- Aspectos conceptuales: describen el contenido multimedia desde el punto de vista de la semántica del mundo real y las nociones conceptuales. Incluyen entidades como objetos, eventos, conceptos abstractos y relaciones.

Los aspectos conceptuales y estructurales están relacionados por un conjunto de enlaces, que permiten al contenido multimedia ser descrito partiendo de ambos aspectos. Los enlaces relacionan diferentes conceptos semánticos con las instancias del contenido que se encuentra descrito en los segmentos. Muchos de los DS's para descripción y gestión del contenido están unidos y, en la práctica, se incluyen mutuamente en sus respectivas descripciones. Por ejemplo, podemos adjuntar en un segmento información sobre el uso, creación y producción del contenido.

[19] Hay que señalar que el contenido descrito puede estar disponible en diferentes modalidades, formatos, esquemas de codificación, y puede haber varias copias. Por ejemplo, un concierto puede ser grabado en dos modalidades: audio y vídeo (y cada una con diferentes parámetros de codificación, por ejemplo). Así tenemos los conceptos de modalidad, perfil (*profile*) e instancia.

D.7.3.1 Navegación y Acceso

Existen DS's para facilitar la navegación y obtención de contenido multimedia. Esto se consigue definiendo resúmenes, particiones, descomposiciones y variaciones del material.

- *Summaries*: Ofrecen resúmenes breves del contenido multimedia para permitir el descubrimiento, navegación, visualización y audición del contenido multimedia. *Summary* incluye dos tipos de navegación: jerárquica y secuencial. En el modo jerárquico, la información se organiza en niveles, cada uno describiendo el contenido multimedia a diferente nivel de detalle. En general, los niveles más cercanos al inicial (*root*) dan resúmenes más generales, y los más alejados, resúmenes más detallados. El resumen secuencial da muestras de vídeo o una secuencia de imágenes, posiblemente sincronizadas con audio, que componen una presentación o visualización rápida.

- Views, Partitions and Decompositions*: Describen diferentes descomposiciones de señales multimedia en el espacio, el tiempo y la frecuencia. Pueden ser usadas para describir diferentes perfiles de los datos multimedia, importantes para el acceso a múltiples resoluciones y para la obtención progresiva del contenido multimedia.*

- *Variations*: Dan información sobre las diferentes variaciones posibles de los contenidos, como resúmenes, versiones escaladas, comprimidas y a baja resolución; y versiones con diferentes idiomas y modalidades (audio, vídeo, imagen, texto, etc.). Una de sus funcionalidades es la selección de la variación más apropiada de un contenido multimedia que pueda sustituir al original, si es necesario. El fin que se persigue es adaptar las diferentes posibilidades de los terminales, redes o preferencias de usuario.

D.7.3.2 Organización del Contenido

MPEG-7 ofrece también DS's para organizar y formar colecciones de contenido multimedia. Collection organiza las colecciones de contenido, segmentos, eventos y/o objetos. Permite que cada colección sea descrita como un todo, basándose en propiedades comunes. En particular, diferentes modelos y estadísticas se pueden especificar para caracterizar los valores de los atributos de las colecciones.

D.7.3.3 Interacción con el Usuario

El último conjunto de herramientas de descripción se encarga de la interacción con el usuario.

El elemento User Preference describe las preferencias del usuario que está haciendo uso del material multimedia. Esto permite, por ejemplo, emparejar las preferencias y la descripción del contenido para facilitar la personalización del acceso, presentación y uso de la información audiovisual.

El otro elemento que contiene información sobre la interacción con el usuario es Usage History, en el que se registran las acciones que lleva a cabo el usuario dentro del sistema.

E XML (eXtensible Markup Language)

E.1 Introducción

XML (*eXtensible Markup Language*) o lenguaje de etiquetado extensible es una recomendación de *W3C* de *1998* [REFERENCIA CURZADA]. Es un lenguaje de especificación de información (relativamente nuevo) que se está convirtiendo en un importante protagonista en muchas aplicaciones *web*. Su aspecto es similar al de *HTML* (es un lenguaje basado en etiquetas). Se define como un lenguaje para crear lenguajes de forma que el programador siempre podrá crear su propia sintaxis para cada caso particular en función de sus necesidades.

Este lenguaje tiene dos importantes características: se trata de un metalenguaje, un lenguaje que permite crear otros lenguajes para definir contenidos, independientemente del tipo de contenido o finalidad que pueda tener, y permite compartir la información con cualquier formato que la necesite. Con las herramientas adecuadas, un *website* puede estar disponible para ser visitado mediante un navegador web y además, puede ofrecer su información a otros sistemas como teléfonos *WAP*, dispositivos *PDA's*, etc.

Un procesador *XSL*[20] (*Extensible Style Language*, Lenguaje de estilo extensible) es un estándar para la visualización de documentos *HTML*. Permite transformar un documento *XML* en uno *HTML*, de texto o en otro tipo de documento *XML*. La especificación está recogida en el *W3C* como *XSL Transfomation* (*XSLT*), el *XML Path Language* (*XPath*) y *XSL-FO* (*XSL Formatting Objects*).

La principal característica de *XML* es que permite realizar procesos de intercambio entre un sistema y otro totalmente diferentes.

E.2 XML Namespaces

XML permite a cada programador o desarrollador crear lenguajes propios de etiquetas para los propios proyectos. Estos lenguajes pueden ser compartidos en diferentes proyectos por todo el mundo. El problema surge cuando se mezclan varios de estos lenguajes XML propios en los que es posible que los elementos tengan el mismo nombre, aunque se refieran a elementos con una funcionalidad completamente distinta, con el consiguiente e inevitable conflicto. La solución es

[20] XSL viene de eXtensible Stylesheet Language (lenguaje extensible de hojas de estilos). Sin embargo, el término se utiliza a menudo tanto en sentido genérico como específico, con lo que lleva a confusión. En general, XSL se refiere a una familia de tres recomendaciones producidas por el grupo de trabajo de XSL del W3C, que son: XSL Transformations (XSLT), XML Path Language (XPath) y eXtensible Stylesheet Language (XSL en sentido estricto). Para evitar la confusión a menudo se refiere a esta última recomendación como XSL-FO (XSL Formatting Objects). XSLT y XPath son recomendaciones del W3C desde noviembre de 1999 y XSL (XSL-FO) desde octubre de 2001.

declarar varios *namespaces* o espacios de nombres, uno por cada lenguaje. Permiten a cada elemento y atributo de un documento XML ser situado en un espacio de nombre diferente. De esta forma pueden coexistir en un mismo documento varios conjuntos de etiquetas.

Namespaces in XML es una recomendación del W3C que fija las bases para que se pueda modularizar estos lenguajes y definir inequívocamente a que lenguaje XML específico se refiere un elemento o atributo.

Un *namespace* es una colección de nombres, identificados por una referencia URI, que se utilizan en documentos XML como tipos de elemento y nombres de atributo. Dos referencias URI se consideran iguales cuando son idénticas carácter a carácter.

Los *namespaces* se definen con el atributo de nombre *xmlns:prefijo* en los elementos en los que se desea que sea aplicado. El valor del atributo es la referencia URI del *namespace*. Cada vez que vaya a ser referido un elemento o atributo de ese *namespace* se debe indicar anteponiendo el prefijo declarado de la siguiente forma: <*prefijo:elemento*>.

Por ejemplo, una *stylesheet XSLT* comienza declarando el *namespace* correpondiente en el nodo raíz:

```
<xsl:stylesheet xmlns:xsl=”http://www.w3.org/XSL/Transform/1.0”>
```

donde se indica el *namespace* http://www.w3.org/XSL/Transform/1.0 asociado al prefijo xsl, y se crea un elemento *stylesheet* perteneciente a ese *namespace*, como indica su prefijo.

A menudo no es tan importante la referencia URI del *namespace* como el hecho de que dentro de un documento se trate de *namespaces* diferentes entre sí. En el siguiente ejemplo se puede ver cómo se declaran y se utilizan múltiples *namespaces*.

```
<DIA xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:MPEG-
7="urn:mpeg:MPEG-7:schema:2001" xmlns="urn:mpeg:mpeg21:dia:schema:2003">
      <Description xsi:type="UserCharacteristicsType">
            <UserInfo xsi:type="MPEG-7:PersonType">
                  <MPEG-7:Name>
                        <MPEG-7:GivenName>Luis</MPEG-7:GivenName>
                        <MPEG-7:FamilyName>Herranz</MPEG-7:FamilyName>
                  </MPEG-7:Name>
            </UserInfo>
      </Description>
</DIA>
```

Como se puede ver, en este fragmento de documento XML (de una descripción real utilizada en el proyecto) el uso de múltiples namespaces permite combinar elementos de varios lenguajes XML diferentes, sin peligro de conflictos entre ellos. En este ejemplo se tienen los siguientes lenguajes, elementos y *namespaces*:

- *DIA, Description, UserInfo y UserCharacteristicsType* pertenecen al *namespace* definido por defecto (cuando se define un *namespace* por defecto todos los elementos que no tienen prefijo se asume que pertenecen a ese *namespace*) y referido a la URI *urn:mpeg:mpeg21:dia:schema:2003,* y que se refiere al esquema de Digital Item Adaptation (MPEG-21 Part 7,).

- El atributo *xsi:type* pertenece al *namespace xsi* referido en la URI http://www.w3.org/2001/XMLSchema-instance, que se refiere a uno de los definidos por la recomendación *XML Schema* en concreto al *namespace* para definir instancias.

- *MPEG-7:PersonType, MPEG-7:name, MPEG-7:GivenName* y *MPEG-7:FamilyName* pertenecen al *namespace* MPEG-7 referido en la URI *urn:MPEG-7:schema:2001* y que se refiere al esquema de MPEG-7.

E.3 XML Schema

El estándar XML Schema es un lenguaje XML para describir y establecer qué pueden contener los documentos XML, estableciendo la estructura que deben tener.

XML Schema viene a cubrir en gran parte la misma funcionalidad que cubrían los DTDs, especificados en el estándar XML 1.0, aunque tratando de resolver muchas de las limitaciones que tienen éstos. El objetivo de los DTDs es definir los bloques legales que pueden construir cualquier documento basado en SGML (Standarized General Markup Language), y como tales existen desde los años 70.

El lenguaje XML Schema fue creado por el W3C basado en diferentes contribuciones de una gran variedad de compañías y particulares, por lo que está diseñado para poder ser utilizado en un amplio rango de situaciones. La recomendación consta de tres partes:

- **XML Schema Part 0: Primer**. Esta primera parte es descriptiva, basada en ejemplos, que introduce algunas de las características claves de XML Schema, y de qué es capaz de hacer.

- **XML Schema Part 1: Structures.** La siguiente parte describe como restringir la estructura de los documentos XML (dónde pueden aparecer los elementos, atributos, etc.). Una vez que se ha declarado un elemento o atributo se pueden definir qué puede contener y que valores están permitidos.

- **XML Schema Part 2: Datatypes.** La tercera parte define un conjunto de tipos de datos predefinidos, a los que se puede asociar el contenido de elementos o valores de

los atributos. Además describe las métodos por los cuales se puede controlar la derivación de nuevos tipos de datos a partir de los que ya se han definido previamente.

E.4 Estructura de un fichero XML

Un fichero *XML* tiene similitud con uno *HTML*, la principal diferencia está en que las etiquetas del fichero *XML* definen por sí mismas el tipo contenido.

Un fichero *XML* es un fichero de texto plano. En primer lugar debe tener siempre una primera línea para definir que se trata de un fichero *XML*: `<?xml version="1.0" encoding="ISO-8859-1" ?>`. Además debe de contener una etiqueta raíz de apertura y cierre que engloba el resto de elementos.

XML utiliza la misma forma de escribir las etiquetas que *HTML*: elementos, atributos y valores. Un elemento está formado por una etiqueta y su contenido (si tiene). Los atributos están situados en las etiquetas de apertura.

E.5 Sintaxis básica en XML Schema

A continuación se dará una breve exposición de las reglas y elementos básicos de la sintaxis de XML Schema necesarios para poder entender los esquemas de perfiles diseñados para el proyecto. Para ello se utilizará algunos ejemplos para que sea más fácil su comprensión.

Un documento XML que define un esquema (XML Schema) comienza con el elemento raíz schema que debe pertenecer al namespace http://www.w3.org/2001/XMLSchema. En este caso se va a utilizar el prefijo xsd para el namespace, por generalizar, aunque es bastante habitual no utilizar prefijo y asignar esta URI al namespace por defecto (de hecho ésto es lo que ocurre en los estándares MPEG-7 y MPEG-21).

Mediante el elemento xsd:element se pueden definir elementos en los esquemas, especificando el tipo de datos con el atributo type.

En XML Schema un elemento puede ser de alguno de estos dos tipos de datos: tipos de datos complejos (complex types) y tipos de datos simples (simple types), en función de qué puede contener. Los tipos de datos simples sólo pueden contener texto, aunque se puede limitar qué tipo de texto está permitido que contengan. Por el contrario, los tipos de datos complejos además pueden tener atributos y elementos hijos. Para indicar que un tipo de datos es complejo hay que utilizar el elemento xsd:complexType y para indicar que es un tipo de datos simple se utiliza xsd:simpleType.

En este ejemplo se muestran los elementos de sintaxis descritos hasta este punto:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:element name="CANCION">
            <xsd:complexType>
                  <xsd:sequence>
                        <xsd:element name="TITULO" type="xsd:string"/>
                        <xsd:element name="AUTOR" type="xsd:string"/>
                  </xsd:sequence>
            </xsd:complexType>
      </xsd:element>
</xsd:schema>
```

Como se puede ver, se trata de un esquema muy simple, en el que el elemento raíz CANCION es un tipo de datos complejo que consta de una secuencia de dos elementos, TITULO y AUTOR, que ambos son de un tipo de datos simple llamado string (que está predefinido en XML Schema).

Dentro de XML Schema hay 44 tipos de datos simples predefinidos. Muchos de ellos son similares a los utilizados en los lenguajes de programación. Entre estos tipos de datos predefinidos se encuentran tipos de datos numéricos, de cadenas de caracteres, tipos de datos para manejar fechas y horas y tipos XML.

Hay otro elemento en el ejemplo anterior, xsd:sequence, que permite especificar que los elementos deben agruparse en el orden indicado en el esquema. XML Schema proporciona tres construcciones para especificar cómo se deben agrupar los elementos dentro de un tipo de datos complejo:

- xsd:all agrupa los elementos de forma que cada elemento puede ocurrir como máximo una sola vez, pero el orden no es importante.
- xsd:choice especifica que un elemento cualquiera de los del grupo debería aparecer. También se puede utilizar para decir que entre N y M elementos del grupo podrían aparecer en cualquier orden.
- xsd:sequence especifica que cada elemento en el grupo debe aparecer exactamente una vez y en el orden especificado.

Para especificar el número de veces que está permido a un elemento aparecer en un punto determinado del documento XML se utiliza los atributos minOccurs y maxOccurs, que indican el número mínimo y máximo de instancias de un elemento que pueden aparecer en ese punto. El valor de estos atributos es un entero mayor o igual a cero. Además, maxOccurs puede tener el valor unbounded, que indica que el elemento puede aparecer un número ilimitado de veces.

Otro elemento importante en la sintaxis de XML Schema es el elemento xsd:attribute, que permite declarar atributos. Con el atributo default de este elemento se puede asignar un valor por

defecto al atributo declarado, y con el atributo use se puede especificar la presencia del atributo según su valor:

- required indica que el atributo debe aparecer.
- optional indica que puede aparecer y puede no hacerlo.
- prohibited indica explícitamente que el atributo no debe aparecer.

En el ejemplo mostrado anteriormente, el tipo de datos complejo del elemento CANCION se define dentro de la propia declaración del elemento. Ésto se conoce como un tipo de datos anónimo, ya que no se ha definido un identificador para ese tipo de datos. Para poder hacer uso de las posibilidades que XML Schema tiene para crear nuevos tipos combinando los que ya se encuentran definidos y los mecanismos de herencia y derivación de tipos hay que definir un nombre para estos tipos. En el siguiente ejemplo se muestra una estructura similar a la del anterior, pero definiendo el tipo de datos:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:element name="CANCION" type="CancionType"/>
      <xsd:complexType name="CancionType">
            <xsd:attribute name="tipo" type="tipoType" use="optional"
default="original"/>
            <xsd:sequence>
                  <xsd:element name="TITULO" type="xsd:string"/>
                  <xsd:element name="AUTOR" type="xsd:string"
minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
      </xsd:complexType>
</xsd:schema>
```

En este ejemplo el elemento CANCION deja de ser de un tipo de datos anónimo para ser del tipo de datos complejo CancionType. Ahora el elemento AUTOR puede no aparecer en el documento XML instancia, o puede aparecer un número ilimitado de veces, mientras que TITULO debe aparecer una y sólo una vez. Además se ha añadido el atributo tipo, que es del tipo de datos simple tipoType, con un valor por defecto de valor original y que su aparición en el documento instancia es opcional.

E.6 Fichero DTD

Un *DTD* (*Document Type Definition*, Definición del tipo de documento) es una definición que establece cómo debe de ser un fichero *XML* que haga uso de este *DTD*. Esta definición hace referencia a los elementos, atributos y entidades que se pueden utilizar así como sus restricciones. Estos ficheros *DTD* son también ficheros *XML* con una sintaxis especial. Definen qué elementos son obligatorios u opcionales, qué atributos son necesarios, qué tipo de datos deben contener, etc.

E.7 Limitaciones de los DTDs y características de XML Schema

XML es un subconjunto de SGML y de él hereda los DTDs. Estos DTDs están pensados fundamentalmente para describir documentos narrativos, es decir, libros, artículos, manuales técnicos, páginas web, etc. Sin embargo, el uso de XML se ha extendido más allá de este tipo de funcionalidades (que en principio son para las que estaba pensado el propio SGML), como pueden ser la serialización de objetos, comercio electrónico, gráficos vectoriales, etc. Por ello, en estas nuevas aplicaciones los DTDs tienen unas importantes limitaciones, entre las cuales destacan:

- La carencia casi absoluta de tipificación de datos, en especial para los elementos. Por ejemplo, los DTDs no pueden indicar que el elemento MES debe contener un número, y mucho menos un número entero entre 1 y 12. Tampoco se puede indicar que el elemento NOMBRE debe contener entre 1 y 255 caracteres. En documentos narrativos para los que estaba pensado SGML no es algo importante, pero para las nuevas aplicaciones de XML sí.

- Los DTDs no tienen sintaxis XML. Esto implica que se necesiten diferentes parsers y APIs para manejar los DTDs y los documentos XML.

- Los DTDs son extensibles de forma muy limitada, y demasiado compleja y artificiosa. Es difícil combinar DTDs independientes de una forma sencilla.

- La compatibilidad con los *namespaces* es muy limitada. Rigurosamente, en los *namespaces* lo importante es la referencia URI. El prefijo que se utilice da igual, ya que en cada documento se puede utilizar uno diferente, siempre que se refiera a la misma URI. La validación de documentos que utilicen prefijos de *namespace* funciona sólo si se declaran en el DTD. No se puede utilizar las referencias URI de los *namespaces* en un DTD.

XML Schema trata de solventar estas limitaciones definiendo una sintaxis nueva basada en XML para describir los posibles contenidos de un documento XML. Esta nueva sintaxis incluye:

- Amplias posibilidades de declaración de tipos de datos.

- Validación de documentos con varios namespaces basada en las URIs en lugar de los prefijos.

- Extensibilidad y escalabilidad.

- Mecanismos para incluir documentación embebida.

- Mecanismos para permitir herencias para las definiciones de elementos, atributos y tipos de datos.

F Funciones Gamma

El propósito de este apéndice es aclarar las posibles dudas matemáticas que puedan surgir al lector después de leerse el desarrollo de algunos algoritmos expuestos en la memoria del PFC.

Para la implementación de dichas funciones usaremos el libro *"Numerical Recipes in C – The art of Scientific Computing 2nd Edition"* [20] en el que se nos explica paso a paso como implementar todo el software necesario para poder calcular dichas funciones trascendentales.

A continuación se pasan a explicar las siguientes funciones: función gamma y funciones gamma incompletas.

F.1 Función Gamma

La función gamma es una función que extiende el concepto de factorial a los números complejos. La notación fue ideada por Adrien-Marie Legendre. Dicha función esta definida por la siguiente integral:

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t}\, dt$$

Cuando el argumento z es un entero, la función gamma entonces es justo la función más conocida como factorial, pero con una pequeña modificación.

$$\Gamma(n+1) = n!$$

para todo número n positivo

Mediante la integración por partes, se puede demostrar la siguiente relación de recurrencia:

$$\Gamma(n+1) = n!$$

También de la misma relación se sigue que

$$\lim_{z \to 0^+} \Gamma(z) = \lim_{z \to 0^+} \frac{\Gamma(z+1)}{z} = \infty$$

A través de la relación

$$\Gamma(1-z)\Gamma(z) = \frac{\pi}{\sin \pi z}$$

válida para todo $z \notin \mathbb{Z}$, se puede hacer una extensión analítica de Γ(z) a todo el plano complejo.

La siguiente forma de definir la función gamma es válida para todos los números complejos excepto para los enteros no positivos:

$$\Gamma(z) = \frac{e^{-\gamma z}}{z} \prod_{n=1}^{\infty} \left(1 + \frac{z}{n}\right)^{-1} e^{z/n}$$

donde γ es la constante de Euler-Mascheroni.

Una forma alternativa de definir la función gamma es:

$$\Gamma(z) = \lim_{n \to \infty} \frac{n! n^z}{\prod_{k=0}^{n} (z+k)}$$

El valor más conocido, para un número no entero, de la función gamma es:

$$\Gamma\left(\frac{1}{2}\right) = \sqrt{\pi}$$

Actualmente existen una gran variedad de métodos numéricos para calcular la función gamma, pero ningún es tan exacto como la aproximación encontrada por Lanczos.

Dicha aproximación nos servirá para poder calcular numéricamente la función gamma y asi poder programarla para realizar su calculo con una computadora. Para unos ciertos números enteros γ y N, y para unos ciertos coeficientes c1, c2, . . . , cN, la función gamma se puede aproximar por

$$\begin{aligned}\Gamma(z+1) &= (z+\gamma+\tfrac{1}{2})^{z+\frac{1}{2}} e^{-(z+\gamma+\frac{1}{2})} \\ &\quad \times \sqrt{2\pi}\left[c_0 + \frac{c_1}{z+1} + \frac{c_2}{z+2} + \cdots + \frac{c_N}{z+N} + \epsilon\right] \quad (z > 0)\end{aligned}$$

de esta aproximación podemos destacar varias cosas.

- La constante c_o esta muy próxima al valor 1
- El termino de error esta parametrizado por Є

- Para un cierto valor $\gamma = 5$, N = 6, y un conjunto de c's, el error es menor que $|Є| < 2$ x10-10.

F.2 Funciones gamma incompletas

Como se ha visto en el apartado anterior, la función gamma esta definida por una integral definida. Ahora pasaremos a ver las funciones gamma incompletas que están definidas por una integral indefinida sobre el mismo integrando.

Existen dos variedades de la función incompleta gamma, dependiendo de si el límite de la integración superior e inferior es variable.

La primera, conocida como Γ(a,x) se define como

$$\Gamma(a,x) = \int_x^\infty t^{a-1}\, e^{-t}\, dt.$$

La segunda, denotada como γ(a,x) se define como

$$\gamma(a,x) = \int_0^x t^{a-1}\, e^{-t}\, dt.$$

En ambos casos, x es una variable real, con x mayor o igual a cero. El parámetro a es una variable compleja (en la que la parte real ha de ser positiva)

Mediante la integración por partes se puede demostrar que:

$$\Gamma(a+1,x) = a\Gamma(a,x) + x^a e^{-x}$$
$$\gamma(a+1,x) = a\gamma(a,x) - x^a e^{-x}.$$

Donde la función gamma esta definida de la siguiente manera:

$$\Gamma(a) = \int_0^\infty t^{a-1}\, e^{-t}\, dt$$

entonces tenemos

$$\gamma(a,x) + \Gamma(a,x) = \Gamma(a).$$

Operando podemos obtener el siguiente desarrollo,

$$\Gamma(a,x) = (a-1)!e^{-x}\sum_{k=0}^{a-1}\frac{x^k}{k!}$$
$$\Gamma(a,0) = \Gamma(a)$$
$$\Gamma(a) = (a-1)!$$

y

$$\gamma(a,x) \to \Gamma(a) \quad \text{as } x \to \infty.$$

Funciones regularizadas gamma

Existen dos funciones regularizadas gamma, una asociada a cada funcion incompleta gamma:

$$P(a,x) = \frac{\gamma(a,x)}{\Gamma(a)}$$ que tiene los valores limite $P(a,0) = 0$ y $P(a,\infty) = 1$

$$Q(a,x) = \frac{\Gamma(a,x)}{\Gamma(a)} = 1 - P(a,x)$$

Para la posterior programación del algoritmo (numérico) que genere valores de la función para un a y un x dado, se usa el desarrollo siguiente:

$$\Gamma(a,x) = e^{-x}x^a\left(\frac{1}{x+}\ \frac{1-a}{1+}\ \frac{1}{x+}\ \frac{2-a}{1+}\ \frac{2}{x+}\cdots\right) \quad (x>0)$$

que converge rápidamente cuando x es menor que a+1

Para que converja rápidamente para valores de x mayores que a+1 entonces deberemos usar la siguiente aproximación:

$$\Gamma(a,x) = e^{-x}x^a\left(\frac{1}{x+1-a-}\ \frac{1\cdot(1-a)}{x+3-a-}\ \frac{2\cdot(2-a)}{x+5-a-}\ \cdots\right) \quad (x>0)$$

En el libro “Numerical Recipes in C – The art of Scientific Computing 2nd Edition” [20] se nos proporcionan rutinas para calcular P(a,x) y Q(a,x)

Enlaces de interés

- Free Incomplete Gamma Function Calculator
  - http://www.danielsoper.com/statcalc/calc24.aspx
- Free Complemented Incomplete Gamma Function Calculator
  - http://www.danielsoper.com/statcalc/calc24.aspx

G Resultados del análisis de una secuencia de imágenes

G.1 Análisis de los datos generados por la aplicación transmisora

Para la prueba 1 se utilizó la secuencia de 300 imágenes (352x288) “Hall Monitor”.

Como niveles de las descripciones generadas a la salida, se solicitó a la aplicación que generase los datos correspondientes al nivel 1. Además se realizó un análisis con los siguientes parámetros:

- fps = 1 cuadro por segundo
- TAF = 50 segundos
- TAO = 10 segundos
- Dimensiones de la secuencia = 352x288

Los resultados que generó la aplicación, los almacenó en la carpeta “tx/Nivel 1” y estaban ordenados de la siguiente manera (en grupos de 30 cuadros):

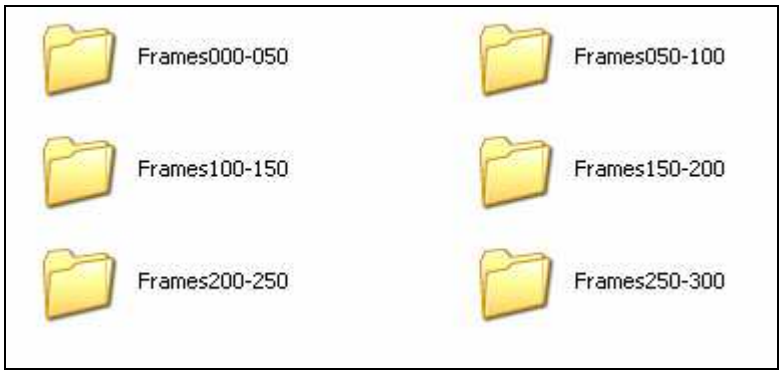


Figura J-1. Análisis realizados a la secuencia original de imágenes

En cada análisis se generó un fondo, una descripción del movimiento y se guardaron las imágenes de cada objeto durante cada división de la secuencia. El contenido se muestra en las siguientes figuras:

Figura J-2. Datos generados del análisis de los cuadros 150-180 de la secuencia original de imágenes

Figura J-3. Imagen de fondo generada del análisis de los cuadros 150-180 de la secuencia original de imágenes

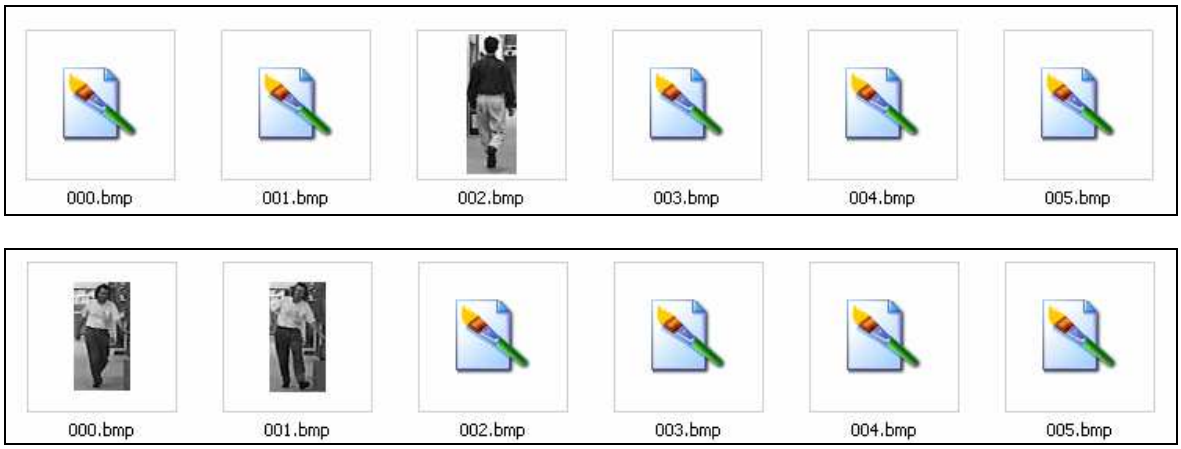


Figura J-4. Objetos extraídos del análisis de los cuadros 150-180 de la secuencia original de imágenes

A continuación se presenta el contenido del fichero que contiene la descripción del movimiento (Description_1.mp7)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Aqui comentarios sobre la aplicacion
-->
<Mpeg7 xmlns="urn:mpeg:mpeg7:schema:2001"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"><DescriptionMetadata>
<Version>1.02</Version><LastUpdate>31-08-2006</LastUpdate>
<Comment>
<FreeTextAnnotation>Esquema de uso general</FreeTextAnnotation>
</Comment><Creator>
<Role href="creatorCS">
<Name>creator</Name>
</Role>
<Agent xsi:type="PersonType">
<Name>
<GivenName>Juan Carlos</GivenName>
<FamilyName>San Miguel Avedillo</FamilyName>
</Name>
</Agent>
</Creator>
<CreationLocation>
<Region>ES</Region>
<AdministrativeUnit>Madrid</AdministrativeUnit>
</CreationLocation>
<CreationTime>2006- 8-18T13:33:40</CreationTime>
</DescriptionMetadata><Description xsi:type="ContentEntityType">
<MultimediaContent xsi:type="ImageType">
<Image>
<MediaInformation>
<MediaProfile master="true">
<MediaFormat>
<Content href="image"/>
<FileFormat href="urn:mpeg:MPEG7FileFormatCS:1"><Name>BMP</Name>
</FileFormat>
<VisualCoding>
<Format href="urn:mpeg:MPEG7FileFormatCS:1" colorDomain="color">
<Name>BMP</Name>
</Format>
<Pixel bitsPer="8"/>
<Frame width="352" height="240"/></VisualCoding>
</MediaFormat><MediaInstance>
<InstanceIdentifier/>
<MediaLocator>
<MediaUri>output\tx\Nivel1\Frames150-200\background200.bmp</MediaUri>
</MediaLocator>
</MediaInstance>
</MediaProfile>
</MediaInformation><VisualDescriptor xsi:type="DominantColorType">
<SpatialCoherency>0</SpatialCoherency>
<Values>
<Percentage>4</Percentage>
<LuminanceValueIndex>255</LuminanceValueIndex>
</Values><Values>
<Percentage>2</Percentage>
<LuminanceValueIndex>143</LuminanceValueIndex>
</Values><Values>
<Percentage>2</Percentage>
<LuminanceValueIndex>148</LuminanceValueIndex>
</Values></VisualDescriptor>
</Image>
</MultimediaContent>
</Description>
<Description xsi:type="ContentEntityType">
<MultimediaContent xsi:type="VideoType">
```

```
<Video>
<SpatioTemporalDecomposition><MovingRegion id="Object1"><TextAnnotation>
<FreeTextAnnotation>Aqui comentarios sobre los objetos</FreeTextAnnotation>
</TextAnnotation><MediaInformation>
<MediaFormat>
<Content href="video" />
<FileFormat href="urn:mpeg:MPEG7FileFormatCS:1"><Name>BMP</Name>
</FileFormat>
<VisualCoding>
<Format href="urn:mpeg:MPEG7FileFormatCS:1" colorDomain="color"><Name>BMP</Name>
</Format>
<Pixel bitsPer="8" />
<Frame width="35" height="86" /></VisualCoding>
</MediaFormat>
<MediaInstance>
<InstanceIdentifier/>
<MediaLocator>
<MediaUri>Frames150-200\Objects\Object001\XXX.bmp</MediaUri>
</MediaLocator></MediaInstance>
</MediaInformation><CreationInformation>
<Creation>
<Title>Creation information</Title>
<Creator>
<Role href="creatorCS"><Name>Creator</Name>
</Role>
<Agent xsi:type="PersonType">
<Name>
<FamilyName>San Miguel Avedillo</FamilyName>
<GivenName>Juan Carlos</GivenName></Name>
</Agent>
</Creator>
</Creation>
</CreationInformation><SpatioTemporalLocator>
<FigureTrajectory type="rectangle">
<MediaTime>
<MediaTimePoint>150</MediaTimePoint><MediaDuration>PT0M50S</MediaDuration>
</MediaTime><Vertex>
<WholeInterval>
<MediaDuration>PT0S</MediaDuration>
</WholeInterval>
<!-- primera dimension--><InterpolationFunctions>
<KeyValue type="startPoint">147</KeyValue>
<KeyValue type="firstOrder">146</KeyValue>
<KeyValue type="firstOrder">143</KeyValue>
<KeyValue type="firstOrder">139</KeyValue>
<KeyValue type="firstOrder">134</KeyValue>
<KeyValue type="firstOrder">107</KeyValue>
</InterpolationFunctions><!-- segunda dimension--><InterpolationFunctions>
<KeyValue type="startPoint">71</KeyValue>
<KeyValue type="firstOrder">68</KeyValue>
<KeyValue type="firstOrder">69</KeyValue>
<KeyValue type="firstOrder">67</KeyValue>
<KeyValue type="firstOrder">67</KeyValue>
<KeyValue type="firstOrder">69</KeyValue>
</InterpolationFunctions>
</Vertex><Vertex>
<WholeInterval>
<MediaDuration>PT0S</MediaDuration>
</WholeInterval>
<!-- primera dimension--><InterpolationFunctions>
<KeyValue type="startPoint">176</KeyValue>
<KeyValue type="firstOrder">177</KeyValue>
<KeyValue type="firstOrder">173</KeyValue>
<KeyValue type="firstOrder">171</KeyValue>
<KeyValue type="firstOrder">165</KeyValue>
<KeyValue type="firstOrder">164</KeyValue>
</InterpolationFunctions><!-- segunda dimension--><InterpolationFunctions>
<KeyValue type="startPoint">71</KeyValue>
```

```
<KeyValue type="firstOrder">68</KeyValue>
<KeyValue type="firstOrder">69</KeyValue>
<KeyValue type="firstOrder">67</KeyValue>
<KeyValue type="firstOrder">67</KeyValue>
<KeyValue type="firstOrder">69</KeyValue>
</InterpolationFunctions>
</Vertex><Vertex>
<WholeInterval>
<MediaDuration>PT0S</MediaDuration>
</WholeInterval>
<!-- primera dimension--><InterpolationFunctions>
<KeyValue type="startPoint">147</KeyValue>
<KeyValue type="firstOrder">146</KeyValue>
<KeyValue type="firstOrder">143</KeyValue>
<KeyValue type="firstOrder">139</KeyValue>
<KeyValue type="firstOrder">134</KeyValue>
<KeyValue type="firstOrder">107</KeyValue>
</InterpolationFunctions><!-- segunda dimension--><InterpolationFunctions>
<KeyValue type="startPoint">150</KeyValue>
<KeyValue type="firstOrder">151</KeyValue>
<KeyValue type="firstOrder">154</KeyValue>
<KeyValue type="firstOrder">156</KeyValue>
<KeyValue type="firstOrder">158</KeyValue>
<KeyValue type="firstOrder">162</KeyValue>
</InterpolationFunctions>
</Vertex><Vertex>
<WholeInterval>
<MediaDuration>PT0S</MediaDuration>
</WholeInterval>
<!-- primera dimension--><InterpolationFunctions>
<KeyValue type="startPoint">176</KeyValue>
<KeyValue type="firstOrder">177</KeyValue>
<KeyValue type="firstOrder">173</KeyValue>
<KeyValue type="firstOrder">171</KeyValue>
<KeyValue type="firstOrder">165</KeyValue>
<KeyValue type="firstOrder">164</KeyValue>
</InterpolationFunctions><!-- segunda dimension--><InterpolationFunctions>
<KeyValue type="startPoint">150</KeyValue>
<KeyValue type="firstOrder">151</KeyValue>
<KeyValue type="firstOrder">154</KeyValue>
<KeyValue type="firstOrder">156</KeyValue>
<KeyValue type="firstOrder">158</KeyValue>
<KeyValue type="firstOrder">162</KeyValue>
</InterpolationFunctions>
</Vertex></FigureTrajectory>
</SpatioTemporalLocator><VisualDescriptor
xsi:type="DominantColorType"><SpatialCoherency>0</SpatialCoherency> <Values>
<Percentage>0</Percentage><LuminanceValueIndex>32</LuminanceValueIndex>
</Values><Values>
<Percentage>0</Percentage><LuminanceValueIndex>36</LuminanceValueIndex>
</Values><Values>
<Percentage>0</Percentage><LuminanceValueIndex>33</LuminanceValueIndex>
</Values></VisualDescriptor></MovingRegion><MovingRegion
id="Object2"><TextAnnotation>
<FreeTextAnnotation>Aqui comentarios sobre los objetos</FreeTextAnnotation>
</TextAnnotation><MediaInformation>
<MediaFormat>
<Content href="video" />
<FileFormat href="urn:mpeg:MPEG7FileFormatCS:1"><Name>BMP</Name>
</FileFormat>
<VisualCoding>
<Format href="urn:mpeg:MPEG7FileFormatCS:1" colorDomain="color"><Name>BMP</Name>
</Format>
<Pixel bitsPer="8" />
<Frame width="40" height="73" /></VisualCoding>
</MediaFormat>
<MediaInstance>
<InstanceIdentifier/>
```

<MediaLocator>
<MediaUri>Frames150-200\Objects\Object002\XXX.bmp</MediaUri>
</MediaLocator></MediaInstance>
</MediaInformation><CreationInformation>
<Creation>
<Title>Creation information</Title>
<Creator>
<Role href="creatorCS"><Name>Creator</Name>
</Role>
<Agent xsi:type="PersonType">
<Name>
<FamilyName>San Miguel Avedillo</FamilyName>
<GivenName>Juan Carlos</GivenName></Name>
</Agent>
</Creator>
</Creation>
</CreationInformation><SpatioTemporalLocator>
<FigureTrajectory type="rectangle">
<MediaTime>
<MediaTimePoint>150</MediaTimePoint><MediaDuration>PT0M50S</MediaDuration>
</MediaTime><Vertex>
<WholeInterval>
<MediaDuration>PT0S</MediaDuration>
</WholeInterval>
<!-- primera dimension--><InterpolationFunctions>
<KeyValue type="startPoint">186</KeyValue>
<KeyValue type="firstOrder">189</KeyValue>
<KeyValue type="firstOrder">189</KeyValue>
<KeyValue type="firstOrder">193</KeyValue>
<KeyValue type="firstOrder">192</KeyValue>
<KeyValue type="firstOrder">188</KeyValue>
</InterpolationFunctions><!-- segunda dimension--><InterpolationFunctions>
<KeyValue type="startPoint">76</KeyValue>
<KeyValue type="firstOrder">75</KeyValue>
<KeyValue type="firstOrder">78</KeyValue>
<KeyValue type="firstOrder">80</KeyValue>
<KeyValue type="firstOrder">78</KeyValue>
<KeyValue type="firstOrder">78</KeyValue>
</InterpolationFunctions>
</Vertex><Vertex>
<WholeInterval>
<MediaDuration>PT0S</MediaDuration>
</WholeInterval>
<!-- primera dimension--><InterpolationFunctions>
<KeyValue type="startPoint">233</KeyValue>
<KeyValue type="firstOrder">234</KeyValue>
<KeyValue type="firstOrder">232</KeyValue>
<KeyValue type="firstOrder">227</KeyValue>
<KeyValue type="firstOrder">228</KeyValue>
<KeyValue type="firstOrder">224</KeyValue>
</InterpolationFunctions><!-- segunda dimension--><InterpolationFunctions>
<KeyValue type="startPoint">76</KeyValue>
<KeyValue type="firstOrder">75</KeyValue>
<KeyValue type="firstOrder">78</KeyValue>
<KeyValue type="firstOrder">80</KeyValue>
<KeyValue type="firstOrder">78</KeyValue>
<KeyValue type="firstOrder">78</KeyValue>
</InterpolationFunctions>
</Vertex><Vertex>
<WholeInterval>
<MediaDuration>PT0S</MediaDuration>
</WholeInterval>
<!-- primera dimension--><InterpolationFunctions>
<KeyValue type="startPoint">186</KeyValue>
<KeyValue type="firstOrder">189</KeyValue>
<KeyValue type="firstOrder">189</KeyValue>
<KeyValue type="firstOrder">193</KeyValue>
<KeyValue type="firstOrder">192</KeyValue>

<KeyValue type="firstOrder">188</KeyValue>
</InterpolationFunctions><!-- segunda dimension--><InterpolationFunctions>
<KeyValue type="startPoint">156</KeyValue>
<KeyValue type="firstOrder">152</KeyValue>
<KeyValue type="firstOrder">152</KeyValue>
<KeyValue type="firstOrder">150</KeyValue>
<KeyValue type="firstOrder">148</KeyValue>
<KeyValue type="firstOrder">147</KeyValue>
</InterpolationFunctions>
</Vertex><Vertex>
<WholeInterval>
<MediaDuration>PT0S</MediaDuration>
</WholeInterval>
<!-- primera dimension--><InterpolationFunctions>
<KeyValue type="startPoint">233</KeyValue>
<KeyValue type="firstOrder">234</KeyValue>
<KeyValue type="firstOrder">232</KeyValue>
<KeyValue type="firstOrder">227</KeyValue>
<KeyValue type="firstOrder">228</KeyValue>
<KeyValue type="firstOrder">224</KeyValue>
</InterpolationFunctions><!-- segunda dimension--><InterpolationFunctions>
<KeyValue type="startPoint">156</KeyValue>
<KeyValue type="firstOrder">152</KeyValue>
<KeyValue type="firstOrder">152</KeyValue>
<KeyValue type="firstOrder">150</KeyValue>
<KeyValue type="firstOrder">148</KeyValue>
<KeyValue type="firstOrder">147</KeyValue>
</InterpolationFunctions>
</Vertex></FigureTrajectory>
</SpatioTemporalLocator><VisualDescriptor
xsi:type="DominantColorType"><SpatialCoherency>0</SpatialCoherency> <Values>
<Percentage>0</Percentage><LuminanceValueIndex>60</LuminanceValueIndex>
</Values><Values>
<Percentage>0</Percentage><LuminanceValueIndex>149</LuminanceValueIndex>
</Values><Values>
<Percentage>0</Percentage><LuminanceValueIndex>61</LuminanceValueIndex>
</Values></VisualDescriptor></MovingRegion>
</SpatioTemporalDecomposition></Video>
</MultimediaContent>
</Description>
</Mpeg7>

PRESUPUESTO

1) Ejecución Material

- Compra de ordenador personal (Software incluido)..... ..... 2.000 €
- Material de oficina ..... 150 €
- Total de ejecución material ..... 2.150 €

2) Gastos generales

- 16 % sobre Ejecución Material ..... 344 €

3) Beneficio Industrial

- 6 % sobre Ejecución Material ..... 129 €

4) Honorarios Proyecto

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

5) Material fungible

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

6) Subtotal del presupuesto

- Subtotal Presupuesto ..... 12010 €

7) I.V.A. aplicable

- 16% Subtotal Presupuesto ..... 1921.6 €

8) Total presupuesto

- Total Presupuesto ..... 13931,6 €

Madrid, Septiembre de 2006

El Ingeniero Jefe de Proyecto

Fdo.: Juan Carlos San Miguel Avedillo
Ingeniero Superior de Telecomunicación

PLIEGO DE CONDICIONES

Este documento contiene las condiciones legales que guiarán la realización, en este proyecto, de un sistema de transmisión de secuencias de vídeo a tasa binaria muy baja y adaptable basada en generación y transmisión de descripciones. 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.