Tesis de Licenciatura en Ciencias de la Computación.

Compresión de secuencias de video de pacientes con epilepsia

Alumno:
Matías Brunstein Macri
(*mbmacri@gmail.com*)

Directora:
Dra. Ana Ruedin

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires, Argentina

15 de octubre de 2011

Resumen

En este trabajo se presenta un método para la compresión de secuencias de video captadas mediante un dispositivo de Video-EEG durante el diagnóstico a pacientes con epilepsia. A diferencia de los métodos de compresión no especializados como MPEG, se aprovechan algunas características propias de este tipo de videos tales como la cámara fija, los movimientos lentos y los largos períodos de inactividad.

Este método de compresión surge de la combinación de diversas técnicas y estrategias comúnmente usadas para la compresión de imágenes y videos. En primer lugar, se realiza una detección de cambios por bloque en base al análisis estadístico de los píxeles a lo largo de un número variable de cuadros consecutivos lo que permite descartar una gran cantidad de información redundante. Sobre la información que aún se conserva se utiliza la técnica de compensación de movimiento para representar parte de los bloques en función del cuadro anterior. Por último, se reduce la entropía de los bloques que deben ser conservados aplicando la *transformada discreta del coseno* conjuntamente a una matriz de cuantización para luego almacenar toda la información mediante un codificador aritmético.

En este trabajo se incluye la explicación detallada de los algoritmos y fórmulas correspondientes a cada técnica utilizada así como también del mecanismo utilizado para estimar los parámetros usados en cada una. También se explica, a través de resultados experimentales, cómo afecta cada técnica utilizada a la calidad y la entropía de los videos. Finalmente, se presenta una comparación entre los resultados obtenidos con este método y los obtenidos al aplicar los estándares de compresión de video MPEG-1 y H.264 a un mismo conjunto de señales de prueba.

A esa pioja que se convirtió
en el amor de mi vida …

Índice

Agradecimientos

A la **Dra. Silvia Kochen**, profesora adjunta de la Cátedra de Neurología de la Facultad de Medicina de la UBA y directora del Centro de Epilepsia del hospital Ramos Mejía, quién originó y motivó este trabajo, y facilitó los videos.

Notación

- *Cuadro k de la secuencia S*: $S^{[k]}$.
- *Subsecuencia entre k y r*: $S^{[k:r]}$.
- *Bloque b del cuadro C*: C_b
- *Bloque b del cuadro k en S*: $S_b^{[k]}$
- *Bloque b en subsecuencia de S*: $S_b^{[k:r]}$
- *Pixel en la posición ij en un cuadro k*: $S_{(i,j)}^{[k]}$ ó $C_{(i,j)}$.
- *Pixel en la posición ij del bloque b en un cuadro k*: $S_{b(i,j)}^{[k]}$.
- *Número de cuadros en una secuencia*: $\#(S)$

1. Introducción

En la actualidad, existe una diversidad de estudios médicos que utilizan como herramienta el monitoreo por video de los pacientes. El monitoreo de pacientes se utiliza con diversos objetivos, desde la simple supervisión remota de un paciente en terapia intensiva, como herramienta de investigación de las funciones cognitivas implicadas en los estudios relacionados con los trastornos del sueño, hasta su uso privilegiado como dispositivo de evaluación diagnóstica de la epilepsia. Este trabajo se centra en un estudio en particular: el Video-Electroencefalograma o *Video-EEG*.

La tecnología de *Video-EEG* [10] es utilizada ampliamente para el diagnóstico de episodios paroxísticos sospechosos de crisis epilépticas y como herramienta indispensable para planificar las cirugías que buscan aliviar esta enfermedad. El Video-EEG consta de un electroencefalógrafo funcionando en sincronismo con una cámara de video que registra al paciente tanto en períodos de reposo como durante las crisis epilépticas. La Figura 1 muestra dos cuadros de estos videos.

Figura 1: Ejemplo de dos secuencias de video producidas por un sistema de Video-EGG.

Estos estudios pueden tomar horas o incluso días. En cada estudio se generan varias horas de videos digitalizados que deben ser conservados para el diagnóstico y el posterior tratamiento del paciente, además, en muchos casos los videos son conservados como historia clínica, para estudios futuros o como material didáctico. Para almacenar toda esta información se requieren varios giga bytes de espacio por estudio, para dar un ejemplo, el *Bio-Logic Digital Video-EGG System* de la firma *Natus Incorporated* genera cerca de 528 megabytes por hora, lo que equivale a 12,7 giga bytes por día [12]. Si bien estos videos luego son editados y recortados manualmente, siguen representando una gran cantidad de información para almacenar.

Usualmente, estos videos son almacenados utilizando algunos de los sistemas estándar de codificación de video tales como MPEG-2, MPEG-4 o incluso H.264, los cuales, si bien tienen muy buenas tasas de compresión, no fueron diseñados

para aprovechar ciertas particularidades que poseen los videos de monitoreo de pacientes. Gracias a estas últimas, se puede aumentar notablemente la tasa de compresión de los mismos.

Este trabajo se enmarca en el campo de la *codificación de videos para el monitoreo de pacientes*. Como se mencionó anteriormente, el almacenamiento (así como la transmisión) de estos videos es un problema creciente. En los últimos años se han publicado varios trabajos al respecto en los cuales se combinan diversas técnicas y estrategias.

Un enfoque utilizado por el estándar MPEG-4 [20] es la representación del video mediante objetos visuales [7][9], lo cual da facilidades para la tarea de edición. Cada objeto se representa como un plano o capa llamada *VOP* (por las siglas en inglés de Video Object Plane), donde cada *VOP* representa una instancia temporal de un objeto de la escena. El contorno del objeto se representa mediante una máscara de bits.

En [19] el autor hace uso de una máscara de bits generada a partir del análisis del desvío estándar de los píxeles a lo largo del eje temporal. En [14] el autor analiza distintas técnicas de segmentación según el movimiento para luego proponer un algoritmo que se basa en la comparación de modelos binarios bidimensionales de los *VOPs* a lo largo de cuadros consecutivos, utilizando la distancia de Hausdorff para segmentar los objetos en movimiento y clasificarlos según el tipo de movimiento (lento o rápido) que tienen los mismos.

En algunos trabajos como [12] y [13] los autores proponen combinar las técnicas de codificación en base a objetos visuales con la detección de cambios mediante la aplicación de dos modelos: uno basado en las cadenas de Markov [27] [23] y otro basado en un test de covarianza el cual aprovecha la correlación temporal entre cuadros consecutivos. Combinando ambas técnicas, los autores crean una estrategia para la construcción de objetos visuales basada en la detección de cambios. En [13] los autores también proponen disminuir el *framerate* (o tasa de cuadros por segundo) en base a la cantidad de movimiento de la escena.

Los autores de [8] plantean que algunas técnicas normalmente utilizadas para estimar movimientos en videos pierden eficiencia cuando los datos no responden a distribuciones paramétricas (como la distribución Normal o de Laplace). Ellos proponen una solución a este problema al utilizar una técnica no paramétrica basada en la entropía del error de estimación.

En [1] se aborda un problema que tiene muchos aspectos en común con el monitoreo de pacientes: la compresión de videos de seguridad. En este caso los autores también proponen la compresión basada en objetos visuales, identificando los objetos en movimiento y codificándolos como referencias a los cuadros anteriores. Cuando la forma del objeto no es rectangular, se utiliza, en vez de la DCT por bloques, una versión de la misma que se ajusta a cualquier forma: SA-DCT (*Shape-Adaptative DCT*)[6].

Este trabajo pretende ofrecer una alternativa simplificada para el almacenamiento de los videos producidos por los sistemas de Video-EEG, aprovechando algunas características propias de este tipo de video. Para ello se analizaron 6 videos de muestra provenientes de estudios reales a fin de determinar qué aspectos de los mismos pueden ser utilizados para mejorar las tasas de compresión. Luego, se planteó un mecanismo automático que permite aislar la información relevante de cada cuadro del video a fin de codificarla descartando todo aquello que resulte insustancial para el estudio médico. Finalmente se integró dicho mecanismo en un algoritmo de codificación (o *CODEC*) sencillo para mostrar su potencial.

Este informe se organiza de la siguiente manera: en la sección 2 se explica brevemente qué es y cómo se trata la epilepsia, y qué rol juega el Video-EGG; en la sección 3 se definen algunos conceptos que serán utilizados a lo largo de este informe; en la Sección 4 se analizan los videos y se distinguen las características de los mismos que se utilizarán para mejorar las tasas de compresión; en la sección 5 se describe la estrategia utilizada para comprimir las secuencias de video; en la sección 6 se presenta una estrategia para identificar qué partes de cada cuadro aportan información relevante para el estudio y qué partes no; las secciones 7 y 8 describen las técnicas de compensación de movimiento y de codificación utilizada; en la sección 9 se muestran los resultados obtenidos y en la sección 10 se exponen las conclusiones.

Finalmente se incluyen 4 apéndices en los que se muestran los algoritmos implementados (Apéndice A), las imágenes y gráficos resultantes del análisis de los resultados obtenidos (Apéndice B), se explican algunos detalles de la implementación (Apéndice C) y, finalmente, se explica brevemente qué pasos se deben seguir para decodificar los videos comprimidos (Apéndice D).

2. Información médica y procedencia de los videos analizados

De las afecciones neurológicas la epilepsia es una de las más frecuentes: se considera que existe una prevalencia que oscila entre el 4 al 10 por 1000 en nuestro país. Como menciona en [4]: "El cerebro es una estructura altamente compleja, compuesto por millones de células nerviosas llamadas neuronas. Generalmente, su actividad está organizada y posee mecanismos de autorregulación. Las neuronas son responsables de una amplia gama de funciones, incluyendo la conciencia, los movimientos y las posturas corporales, por lo que una repentina interrupción temporaria en alguna o todas estas funciones, puede ser llamada crisis. Esta puede ser causada por desórdenes crecientes dentro del cerebro (causa intrínseca) o más raramente, por un factor externo como la falta temporaria de oxígeno o glucosa. No siempre resulta fácil dar una explicación para cada caso, sobre la causa de la crisis. Las epilepsias, afectan a diferentes personas de maneras diversas."

"Si bien el cerebro de toda persona puede generar una crisis si existen ciertas condiciones, la mayoría de los cerebros tiene alta resistencia a las crisis [...] Una persona con baja resistencia podría desarrollar la epilepsia espontáneamente, sin que otros factores estén involucrados."

"Los síntomas que ocurren durante una crisis epiléptica son transitorios y no presentan secuelas físicas. El diagnóstico está basado en una historia que cuenta con más de una crisis epiléptica. Un testigo ocular preciso es crucial para hacer un correcto diagnóstico, tanto o más que la persona que experimenta la crisis, que no registra lo sucedido. La realización de estudios diagnósticos va a dar información adicional sobre la epilepsia: electroencefalogramas -EEG-, Video-EEG, resonancia magnética nuclear, dosaje en sangre de la medicación antiepiléptica."

La respuesta al tratamiento varía según las personas. El 64 % de los pacientes controla bien sus crisis, en la mayoría de los casos con una sola droga, sin embargo según la Dra. Silvia Kochen (quien originó y motivó este trabajo) existe un 30 % de casos denominados *epilepsia refractaria*, los cuales no responden a medicamentos. Esto afecta significativamente la calidad de vida de los enfermos, porque las crisis pueden sobrevenir en cualquier momento. Pero en la mitad de los casos es posible que el paciente refractario mejore con una cirugía.

Claro está que una cirugía cerebral es altamente riesgosa, por lo tanto, antes de realizar cualquier tipo de intervención se debe tener un mapa extremadamente preciso del cerebro del paciente y, fundamentalmente, de la localización de la zona donde se produce la epicrisis. Según comenta la Dra. Kochen en una nota periodística [15]: "En el 85 % de los casos las epilepsias se originan en el lóbulo temporal, es decir, allí se encuentra la zona epileptógena y a partir de ese lugar se propaga el desorden neuronal característico de la enfermedad. Sin embargo, es

crucial diagnosticar el lugar preciso del cerebro donde se inicia la crisis epiléptica."

El procedimiento más aceptado a nivel mundial para detectar el punto de origen de las crisis es el *videoelectroencefalógrafo* (o Video-EEG) que consiste en un equipo de EEG computarizado, equipado con una videocámara que registra las ondas cerebrales en sincronismo con un video del paciente. La implementación es idéntica al electroencefalograma tradicional, al paciente se le colocan varios electrodos sobre el cuero cabelludo, la computadora registra cada instancia de su actividad cerebral durante varias crisis a excepción de que esta vez se agrega la captura de imágenes a fin de comparar las manifestaciones físicas con el patrón electroencefalográfico. Este estudio puede extenderse durante algunas horas o varios días.

"El paciente está muy tranquilo, en una habitación, acompañado por el técnico y la enfermera, que son profesionales importantísimos en esta tarea. El gran problema es que la señal cerebral tiene mucho ruido: alguien pestañea, mastica o abre los ojos y en la pantalla aparecen imágenes confusas. Por eso es tan importante poder registrar qué hace el paciente mientras se registran las ondas cerebrales. Así es posible hallar un patrón de señales que, acompañado por los síntomas que la persona describe, puede indicarnos con bastante precisión dónde comienza la crisis. Muchas veces tienen una señal electroencefalográfica quizá no tan impresionante como las que producen los artificios musculares, que hay que saber diferenciar. Lo más importante son los diez primeros segundos, que dan el patrón de señales."

El diagnóstico se completa con una resonancia magnética y se decide si el paciente es buen candidato a una cirugía. "Lo que se hace es una lobectomía temporal. Se ablaciona una parte de la zona del cerebro donde se produce esa descarga hipersincrónica de una población neuronal y de esa manera disminuyen las crisis".

3. Algunos conceptos previos

Esta sección tiene como objetivo introducir los conceptos y definiciones necesarios para entender este trabajo. También se describen algunas operaciones sobre videos y cuadros que se utilizan más adelante.

3.1. Videos, secuencias y cuadros

Un video es una secuencia de cuadros, y cada cuadro una matriz de valores enteros en el rango [0-255] si es en escala de grises o 3 matrices (una para cada componente de color) si es en *RGB*. Sin embargo, durante la mayor parte del trabajo se utilizan videos en escala de grises, de modo que, salvo que se aclare lo contrario, siempre se hará referencia a cuadros en escala de grises.

Todos los videos usados en este trabajo tienen el mismo tamaño de cuadro: 240 píxeles de alto por 352 píxeles de ancho, de modo que cada vez se hace referencia al cuadro C se trata de una matriz de valores enteros de 249×352 componentes.

A su vez, a los cuadros se los divide en bloques no solapados de 8×8 píxeles. Para hacer referencia a dichos bloques, se numeran por filas tal como se muestra en la figura 2.

0	1	2	3	...	43
44	...	...	...	...	...
...	...	...	...	...	...
...	...	...	...	...	...
...	...	...	...	...	1320

Figura 2: Numeración de los bloques dentro de un cuadro.

3.2. Operaciones entre cuadros

Para simplificar la notación que se utiliza de aquí en adelante, se definen las siguientes operaciones aritméticas entre cuadros o bloques:

***Diferencia*:** Entre dos bloques $B_1 - B_2$ o dos cuadros $C_1 - C_2$. Se calcula como la resta pixel a pixel del primer elemento menos el segundo.

***Suma*:** Entre dos bloques $B_1 + B_2$ o dos cuadros $C_1 + C_2$. Se calcula como la suma pixel a pixel de los bloques o cuadros.

***Media Espacial*:** O simplemente *media* de un bloque $\mu(B)$. Es el valor medio de los 64 píxeles que componen el bloque. Para un bloque B su media espacial se calcula como

$$\mu(B) = \frac{1}{64} \sum_{i=0}^{7} \sum_{j=0}^{7} B_{(i,j)} \tag{1}$$

***Desvío Estándar*:** El desvío de un bloque $\sigma(B)$ es el desvío estándar de los valores de los píxeles que lo componen. Se calcula como

$$\sigma(B) = \sqrt{\frac{1}{64} \sum_{i=0}^{7} \sum_{j=0}^{7} \left(B_{(i,j)} - \mu(B)\right)^2} \tag{2}$$

***Media Temporal*:** del bloque b en los cuadros s a t: $\mu\left(S_b^{[s:t]}\right)$ puede verse como el bloque promedio de una subsecuencia de video en la posición b. El resultado es un bloque $\widetilde{B}$ conformado por el promedio de cada pixel a lo largo de cada cuadro. Dicho formalmente, $\widetilde{B} = \mu\left(S_b^{[s:t]}\right)$ es el bloque tal que

$$\widetilde{B}_{(i,j)} = \frac{\sum_{k=s}^{t} B_{(i,j)}^{[k]}}{t - s + 1} \tag{3}$$

para todo $i, j = 0 \ldots 7$

3.3. Definiciones

Estas son las definiciones de algunos términos que utilizaremos en este documento:

***CODEC*:** Es la abreviatura de codificador-decodificador. Describe una especificación desarrollada en software, hardware o una combinación de ambos, capaz de transformar un archivo con un flujo de datos (*stream*) o una señal. Los *CODECs* pueden codificar el flujo o la señal (a menudo para la transmisión, el almacenaje o el cifrado) y recuperarlo o descifrarlo del mismo modo para la reproducción o la manipulación en un formato más apropiado para estas operaciones.

***Codificación Diferencial*:** Es un método básico utilizado para bajar la entropía de una secuencia de video y consiste en codificar cada cuadro como la diferencia en relación al anterior.

***Suma de Diferencias Absolutas*:** o *SAD* [11][22], es un método para evaluar las diferencias entre dos bloques: Los valores de color de una imagen se comparan pixel a pixel con los de la siguiente y se suman los valores absolutos de las diferencias. El resultado es un número positivo que representa la magnitud de las diferencias punto a punto entre dos imágenes. Dados dos bloques A y B podemos calcular la *suma de diferencias absolutas* como

$$SAD(A, B) = \Sigma_{ij} \left| A_{(i,j)} - B_{(i,j)} \right| \tag{4}$$

***Compensación de Movimiento*:** o *MC* [18] por sus siglas en inglés, es una técnica utilizada en la codificación de vídeo, cuyo principal objetivo consiste en eliminar la redundancia temporal existente entre las imágenes que componen una secuencia, con el fin de aumentar la compresión. El proceso se basa en un algoritmo que examina fotogramas consecutivos, generalmente muy similares entre sí, para analizar y estimar el movimiento entre los dos. Si el sistema detecta que una región de la imagen ya ha aparecido anteriormente, codifica la posición que ocupa en el fotograma actual en lugar de volver a codificar toda la región. De este modo, la predicción de la imagen actual estará determinada por la compensación de movimiento basándose en las imágenes anteriores.

***Entropía*:** o *entropía de shannon* es una medida matemática de la cantidad media de información que contiene una variable aleatoria. Sirve para estimar la cantidad de bits por pixel que se obtiene si se comprime una señal con un compresor basado en la entropía, como Huffman o un codificador aritmético.

La información que aporta un determinado valor, x_i, de una variable aleatoria discreta X, se define como:

$$H(x_i) = \log_2 \frac{1}{p(x_i)} = -\log_2 p(x_i) \tag{5}$$

cuya unidad es el *bit*.

***Entropía de la Codificación Diferencial*:** hace referencia a la entropía de la diferencia entre dos cuadros consecutivos de una secuencia de video. Dado un cuadro $C = S^{[k]}$ se puede calcular la *entropía de la codificación diferencial* de C como:

$$H\left(S^{[k]} - S^{[k-1]}\right) \tag{6}$$

***Tasa de Compresión*:** Es un coeficiente que indica el nivel de compresión alcanzado al comprimir una señal. Se expresa como $n : 1$ donde n es la cantidad de unidades de información de la señal original que se pueden expresar como una única unidad en la señal comprimida.

***MSE*:** Sigla en inglés para el *error cudrático medio*. Es una medida de la distancia entre una señal y una señal aproximada. El *MSE* para la diferencia de dos bloques se calcula con la fórmula

$$MSE(B, \widetilde{B}) = \sum_{i,j} \frac{\left(B_{(i,j)} - \widetilde{B}_{(i,j)}\right)^2}{64} \tag{7}$$

donde B y $\widetilde{B}$ son un bloque y un bloque aproximado.

El *MSE* para la diferencia de dos cuadros se calcula con la fórmula

$$MSE(C, \widetilde{C}) = \sum_{i,j,b} \frac{\left(C_{b(i,j)} - \widetilde{C}_{b(i,j)}\right)^2}{R \times 64} \tag{8}$$

donde C es el cuadro original, $\tilde{C}$ es el aproximado y R es la cantidad de bloques por cuadro.

***PSNR*:** Sigla en ingles para la *relación señal a ruido en el pico* es un término utilizado para definir la relación entre la máxima energía posible de una señal y el ruido que afecta a su representación fidedigna. Debido a que muchas señales tienen un gran rango dinámico, el PSNR se expresa en escala logarítmica, utilizando como unidad el *decibelio*. En el ámbito de la compresión de imágenes se usa como medida cuantitativa de la calidad de la reconstrucción de una imagen (o video). La fórmula para calcularlo es

$$PNSR = 10 \times \log_{10}\left(\frac{MAX_S^2}{MSE}\right) \tag{9}$$

donde MAX_S es el máximo valor posible de S (en el contexto de este trabajo es 255) y MSE es el error cuadrático medio calculado entre S y su recuperada.

***Cuantización*:** Es un proceso de aproximación en el cual los valores de una señal continua (o con un rango amplio de valores) son aproximados por un conjunto reducido de valores discretos. Una de las consecuencias directas de cuantizar una señal es la reducción de su entropía.

***Transformada Discreta del Coseno*:** o *DCT* es una transformada, basada en la *transformada de Fourier*, que permite expresar una secuencia finita de valores reales como el resultado de la suma de distintas señales sinusoidales generadas por la función *coseno* (con distintas frecuencias y amplitudes). La *DCT* es una función biyectiva con dominio y codominio en R^N que puede aplicarse tanto a

vectores como a matrices (de 2 dimensiones o más) y que por sus características es ampliamente utilizada para la compresión de imágenes. En este trabajo notaremos a la función de transformación como $B = DCT(A)$ y a su inversa como $A = IDCT(B)$.

La fórmula para calcular la matriz transformada B de $N \times M$ valores a partir de la matriz A de iguales dimensiones es

$$B_{pq} = \alpha_p \alpha_q \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} A_{mn} \cos \frac{\pi (2m+1) p}{2M} \cos \frac{\pi (2n+1) p}{2N} \tag{10}$$

siendo su inversa

$$A_{nm} = \sum_{p=0}^{q-1} \sum_{n=0}^{N-1} B_{pq} \alpha_p \alpha_q \cos \frac{\pi (2m+1) p}{2M} \cos \frac{\pi (2n+1) p}{2N} \tag{11}$$

donde, para ambos casos, vale que

$$0 \leq p \leq M-1, \quad 0 \leq q \leq N-1 \tag{12}$$

y

$$\alpha_p = \begin{cases} 1/\sqrt{M}, & p = 0 \\ \sqrt{2/M}, & 1 \leq p \leq M-1 \end{cases} \qquad \alpha_q = \begin{cases} 1/\sqrt{N}, & q = 0 \\ \sqrt{2/N}, & 1 \leq q \leq N-1 \end{cases} \tag{13}$$

4. Análisis de las secuencias de video

En líneas generales, los 6 videos estudiados comparten algunas características particulares que se deben a las condiciones en que fueron filmados y al modo en que son utilizados.

Comúnmente los videos son tomados con una cámara de uso hogareño montada sobre un trípode, mientras el paciente se encuentra en reposo en una cama o camilla. La cámara rara vez se mueve (aunque en algunos momentos aislados se observan cambios bruscos de zoom) lo que determina que el fondo de la escena no cambie por largos períodos de tiempo. La habitación se encuentra iluminada con luz artificial que en ocasiones resulta insuficiente. Además, en muchos casos, se utilizan tubos fluorescentes, lo que produce un parpadeo constante que es captado por la cámara.

El ruido que se observa en el video proviene de dos fuentes: por un lado, una cámara de baja calidad y una iluminación inadecuada traen como consecuencia altos niveles de *ruido blanco gaussiano aditivo* [3][26] Además, como los videos originales están grabados en formato MPEG-1, existen errores de reconstrucción que producen distorsiones (pixelados) debido a la cuantización de la DCT por bloques de 8×8.

A los fines médicos resulta significativo destacar que el foco del especialista que estudiará el video (el *observador*) estará puesto exclusivamente en el paciente y no en el resto de los objetos que componen la escena, tales como el fondo, los instrumentos médicos, el ruido, los cambios y oscilaciones en la luminosidad o incluso el color. Consecuentemente solo algunas partes y características de la escena aportan información relevante al *observador*.

4.1. Conjunto de videos de prueba

Este trabajo se basó en el estudio de 6 secuencias de video que fueron utilizadas en tratamientos reales de la epilepsia. Estos videos tienen un tamaño de 240 píxeles de alto por 352 píxeles de ancho, con una profundidad de color de 24 bits por pixel, fueron capturados a 29,97 cuadros por segundo y se encuentran codificados con MPEG-1 [2] a un *bitrate* promedio de $1400 Kbps$. En la Tabla 1 se resumen los aspectos más importantes de cada señal y la Figura 3 muestra algunos cuadros de las mismas.

Para realizar pruebas preliminares y ajustes sobre los algoritmos, en lugar de trabajar directamente con las secuencias completas, durante gran parte de este trabajo se utilizaron 5 fragmentos de 1438 cuadros extraídos de las secuencias antes mencionadas: *P1_EXT_K1 - Fragmento 1*, *P1_EXT_K4 - Fragmento 2*, *P1_EXT_K4 - Fragmento 2*, *P2_EXT2_K2-3 - Fragmento 1* y *P2_EXT2_K2-3 - Fragmento 2*.

Secuencia completa	Duración	Tamaño	Bitrate
P1_EXT_K1	53s	11.921*KBytes*	1.423*Kbps*
P1_EXT_K3	4m 10s	51.558*KBytes*	1.322*Kbps*
P1_EXT_K4	7m 36s	94.405*KBytes*	1.375*Kbps*
P2_EXT2_K2-3	13m 21s	165.250*KBytes*	1.322*Kbps*
P3_EXT8_K2-3	13m 21s	165.256*KBytes*	1.323*Kbps*
P3_EXT10_K4	8m 41s	115.609*KBytes*	1.491*Kbps*

Tabla 1: Características de las señales de prueba.

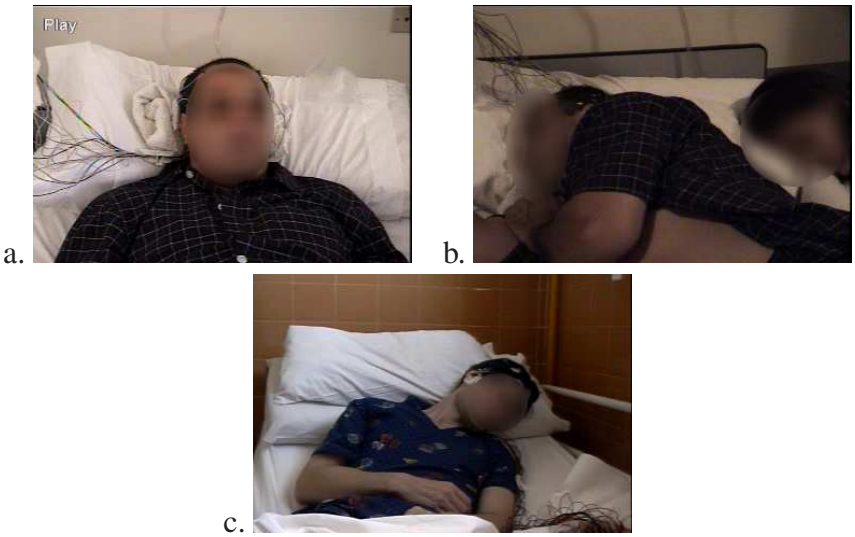


Figura 3: Secuencias utilizadas. (a) *P1_EXT_K1*, (b) *P1_EXT_K4* y (c) *P2_EXT2_K2-3*.

Los resultados finales fueron calculados sobre las secuencias completas y pueden verse en la Sección 9.

Debido a que el color aporta información secundaria, los fragmentos fueron transformados a escala de grises. Esto representa una importante reducción en la cantidad información y simplifica el procesamiento de las secuencias pues sólo se debe trabajar con un canal, a diferencia de los tres canales (R, G y B) que componen una secuencia de video en color.

Luego, con el objetivo de facilitar el proceso de lectura, las secuencias fueron codificadas en formato AVI (*Audio Video Interleave* o *Audio y Video Intercalado*) sin comprimir, de modo que no existe pérdida de información. Para ello se utilizó la herramienta Total Video Converter [1]. Una vez en formato AVI todos los

[1]Total Video Converter v3.12.080330 Copyright ©EffectMatrix Inc. 2004-2008 *http://www.effectmatrix.com*

fragmentos ocupan exactamente 115,9 MBytes.

4.2. Codificación diferencial

Antes de analizar la redundancia de información en las secuencias de video es necesario medir cuánta información hay en cada una de ellas, para ello, se puede analizar la entropía de cada cuadro y la entropía promedio de todos los cuadros. Esto da una estimación de la tasa de compresión que se puede lograr con un compresor basado en entropía. La Figura 4 muestra la entropía cuadro por cuadro de los cinco fragmentos.

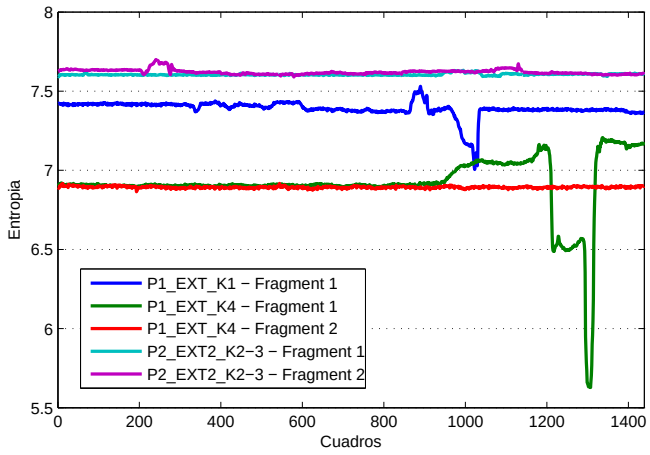


Figura 4: Entropía cuadro a cuadro de los cinco fragmentos de video.

En el gráfico se puede observar que en algunos segmentos la entropía varía más que en otros (en particular la entropía del fragmento *P1_EXT_K4 - Fragmento 1* varía abruptamente entre los cuadros 900 y 1350 debido a cambios abruptos en el zoom de la cámara), sin embargo en todos los casos el promedio se mantiene entre 6,8 y 7,6. Si se considera que la entropía máxima para un video en escala de grises es de 8, no se puede esperar una tasa de compresión muy elevada (los resultados se muestran en la primera parte de la Tabla 2).

El método más sencillo para disminuir la entropía en videos con poco movimiento consiste en codificar cada cuadro como la diferencia respecto al anterior, aplicando la fórmula

$$\begin{aligned} T^{[0]} &= S^{[0]} \\ T^{[n]} &= S^{[n-1]} - S^{[n]} \end{aligned} \tag{14}$$

donde S es nuestra secuencia original y T es la secuencia codificada. Calculando la entropía de T se obtuvieron los resultados que se muestran en la segunda parte de la Tabla 2. A esta técnica se la conoce como *codificación diferencial*.

Fragmento	Codificación Directa		Codificación Diferencial	
	Entropía	T. Estimada	Entropía	T. Estimad
P1_EXT_K1 - Fragmento 1	7,3893	1,08 : 1	3,2846	2,43 : 1
P1_EXT_K4 - Fragmento 1	6,9153	1,15 : 1	2,9748	2,68 : 1
P1_EXT_K4 - Fragmento 2	6,8930	1,16 : 1	2,6750	2,99 : 1
P2_EXT2_K2-3 - Fragmento 1	7,6052	1,05 : 1	2,1952	3,64 : 1
P2_EXT2_K2-3 - Fragmento 2	7,6213	1,04 : 1	2,5084	3,18 : 1
Promedio	7,2848	1,09 : 1	2,7276	2,98 : 1

Tabla 2: Comparación de la entropía y la *tasa de compresión estimada* (abreviada como *T. Estimada*) para los fragmentos utilizando codificación directa y *codificación diferencial*.

Los resultados obtenidos con *codificación diferencial* son sensiblemente favorables a los obtenidos al intentar codificar las señales directamente. Sin embargo la *tasa de compresión estimada* sigue siendo baja, esto se debe al ruido que se encuentra presente en todas las secuencias analizadas.

Si se observan, por ejemplo, los cuadros 146 y 147 de la secuencia *P1_EXT_K4 - Fragmento 1* que se incluyen en la Figura 5, no se aprecian diferencias significativas a simple vista. Sin embargo, al analizar las diferencias pixel a pixel entre ambos cuadros, se nota que un gran número de píxeles (el 49,1 %) presentan distintos valores entre un cuadro y el otro. Incluso se observan muchos cambios en el fondo de la escena, el cual es supuestamente idéntico en ambos cuadros. Como se puede ver en la Tabla 3 el porcentaje de píxeles diferentes es significativo para todos los fragmentos.

Fragmento	Porcentaje de píxeles con cambios
P1_EXT_K1 - Fragmento 1	68.5 %
P1_EXT_K4 - Fragmento 1	64.4 %
P1_EXT_K4 - Fragmento 2	60.7 %
P2_EXT2_K2-3 - Fragmento 1	51.7 %
P2_EXT2_K2-3 - Fragmento 2	53.6 %

Tabla 3: Porcentaje de píxeles que presentan diferencias entre cuadros consecutivos para los fragmentos de video.

Este análisis fue realizado sobre los cuadros completos, por lo tanto, parte de las diferencias se deben a movimientos del paciente. Para poder diferencias aquéllos píxeles que cambian como consecuencia de los movimientos del paciente

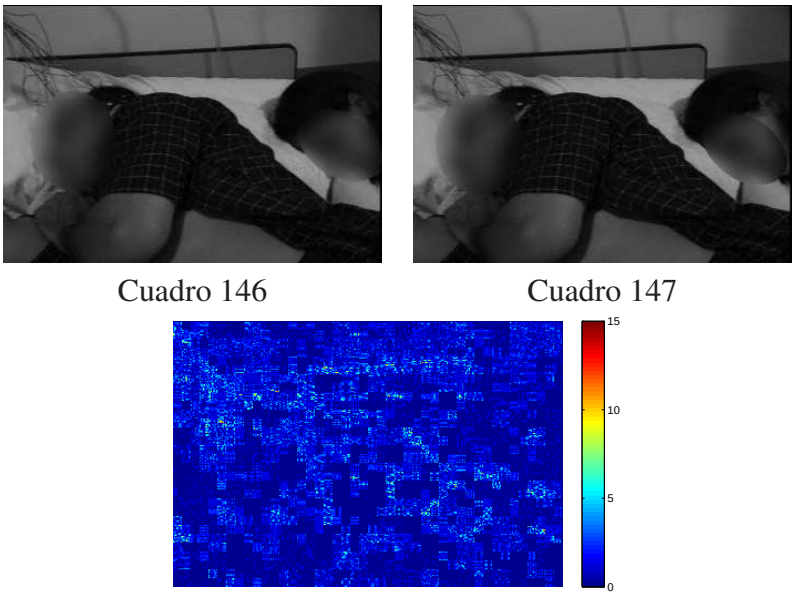


Figura 5: Comparación entre los cuadros 146 y 147 del fragmento *P1_EXT_K4 - Fragmento 1*. La escala de colores del lado derecho permite determinar qué tan grandes son las diferencias. Se nota un pixelado y se ven los errores en la reconstrucción debido a la cuantización de la DCT por bloques de 8x8. La diferencia entre cuadros se debe a las diferencias en los bloques de 8x8.

(o de algún objeto de la escena) y aquellos cuyos cambios se deben al ruido, se determinó analizar las propiedades estadísticas del ruido en los fragmentos de video.

En primer término, se aisló una sección de cada fragmento que no incluye al paciente ni a ningún objeto en movimiento. Al hacer esto se puede asegurar que cualquier diferencia entre cuadros consecutivos se debe exclusivamente al ruido. Luego se calcularon las diferencias pixel a pixel entre cada par de cuadros consecutivos. Los resultados se muestran en la Figura 6.

Si bien se observa un comportamiento gaussiano en las diferencias de los cuadros (pixel a pixel), hay una fuerte correlación espacial. Esto se debe a una mezcla entre el ruido de la cámara y las distorsiones del video provocadas por el compresor MPEG. En efecto, el video original ya viene comprimido en MPEG-1. En la Figura 5 se nota un pixelado y se ven los errores en la reconstrucción debido a la cuantización de la DCT por bloques de 8×8. Aparentemente, el pixelado domina al ruido de la cámara, y no se pueden separar.

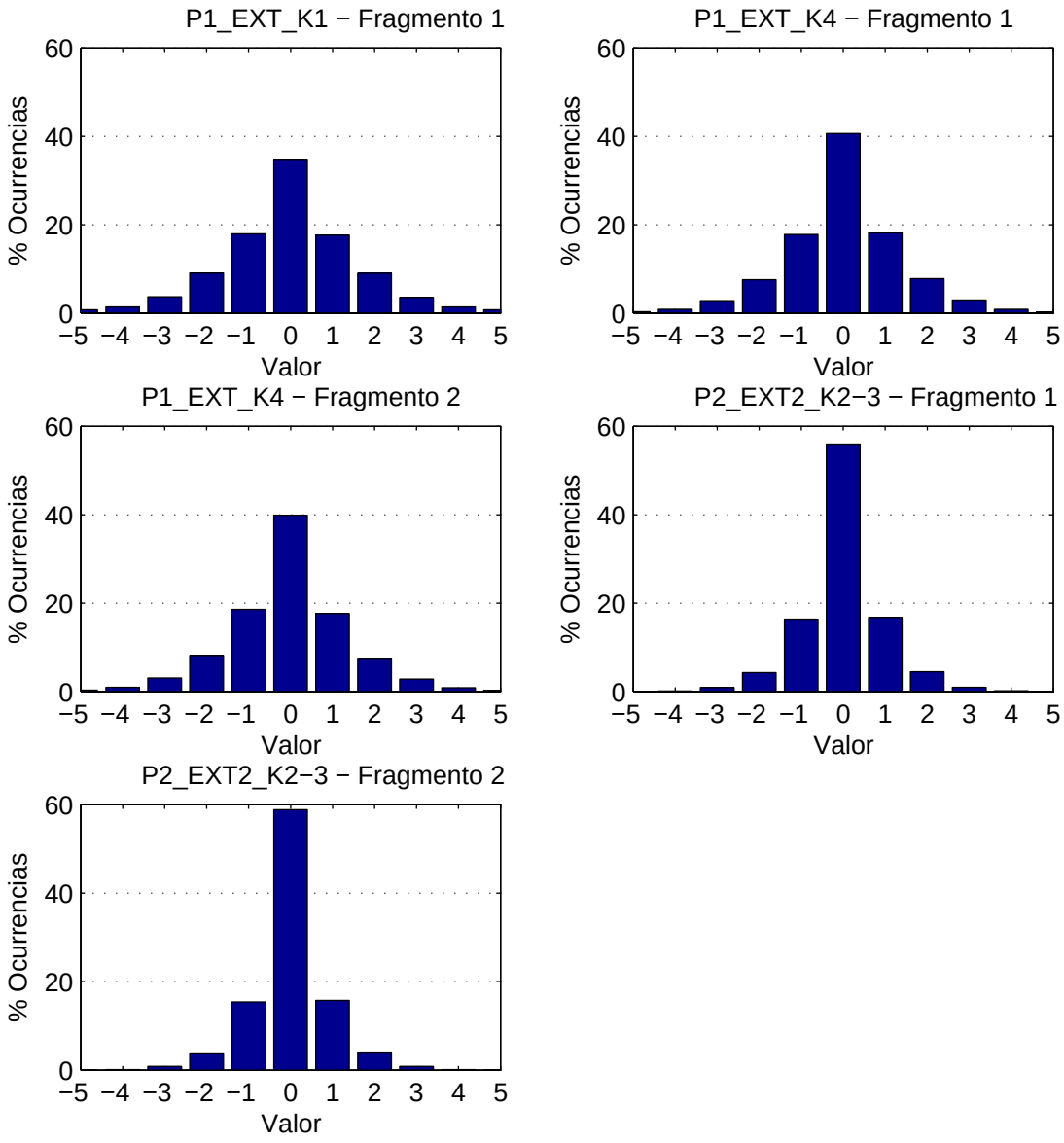


Figura 6: Histogramas de las diferencias de dos cuadros en una región sin movimiento de cada uno de los fragmentos.

Escenario	Tasa de Compresión
Sin codificar sin agregar ruido	1,07 : 1
Sin codificar agregando ruido	1,07 : 1
Usando *Cod. Dif.* sin agregar ruido	27,55 : 1
Usando *Cod. Dif.* agregando ruido	2,972 : 1
Usando *Cod. Dif.* agregando ruido y filtrando con Wiener	4,54 : 1

Tabla 4: Tasa de compresión estimada para una secuencia de 50 cuadros de 512×512 píxeles bajo distintas condiciones de codificación y ruido.

Para calcular cómo impacta el *ruido* al aplicar la *codificación diferencial* a un video, se realizó un sencillo experimento: se generó una secuencia de video de 50 cuadros de 512×512 en escala de grises de 8 bits, en la que la escena no cambia en ningún momento. Luego calculamos la *tasa de compresión estimada* sobre la secuencia bajo distintas condiciones de codificación y ruido (el ruido agregado responde a una distribución Normal $N(\mu = 0, \sigma = 0{,}91)$). Los resultados se muestran en la Tabla 4.

Los valores en la Tabla 4 dejan en evidencia que el ruido presente en las secuencias de video hace inviable el uso directo de la *codificación diferencial* como estrategia de compresión. El quinto experimento incluido en la tabla consistió en aplicar el filtro de Wiener [25] con vecindad de 3×3 a cada cuadro antes de aplicar la *codificación diferencial*, pero aún así los resultados no fueron favorables.

En los capítulos sucesivos se describe el método utilizado para comprimir estos videos haciendo uso de alguna de las propiedades descriptas en esta sección.

5. Estrategia del método

Como se vió en la sección anterior, las señales estudiadas comparten un conjunto de características que pueden aprovecharse para diseñar una estrategia de compresión optimizada para videos producidos por equipos de Video-ECC. Para precisar, este método de compresión explota las siguientes características:

- Los videos son filmados con una cámara fija, montada sobre un trípode, la que rara vez se mueve.
- El fondo de la escena tiende a ser estático y la mayor parte de los cambios se deben al ruido de la cámara sumado a las distorsiones de la compresión del video por MPEG-1.
- En general el movimiento presente en la escena es generado por el paciente.
- Hay largos periodos de inactividad.
- Los colores no aportan información relevante para el diagnóstico.

La Dra. Silvia Kochen, señala que el tipo de movimiento que realiza el paciente durante las crisis es de valiosa ayuda para determinar la zona a intervenir quirúrgicamente. Es por esto que este método de compresión debe enfocarse en conservar toda la información posible de las regiones que contienen movimiento, pudiendo descartar mucha información que resulta insustancial para el diagnóstico médico.

En este trabajo se propone un método de compresión con perdida de información con bases en el estándar MPEG [2] orientado al almacenamiento de secuencias de video producidas por los sistemas de Video-ECC. Como en todo algoritmo con pérdida sólo se conserva una porción de la información total del video, se puede decir que la información relevante para el diagnóstico será *codificada* mientras que la información irrelevante será *descartada*.

El diagrama de bloques que se muestra en la Figura 7 es una representación gráfica de las distintas etapas que componen al método de compresión presentado en este trabajo. En cada etapa se aplica una transformación a la señal de video y, en algunos casos, se cambia su representación.

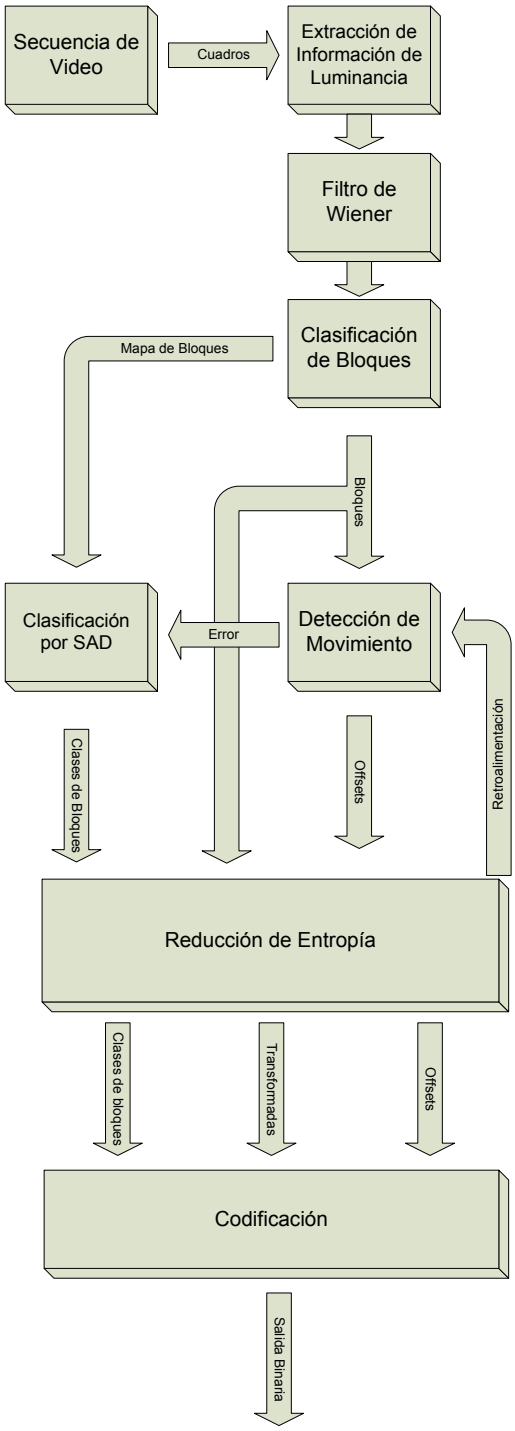


Figura 7: Diagrama de bloques que describe la estrategia de compresión propuesta en este trabajo.

El primer bloque, llamado “Secuencia de Video” representa la secuencia de video a ser comprimida, la cual ingresa al sistema cuadro por cuadro. El video original es en colores, lo primero que se hace es extraer la información de luminancia de cada pixel para obtener así un video en escala de grises. Para obtener la luminancia de un pixel *RGB* se aplica la siguiente transformación [16]:

$$I = \begin{bmatrix} 0{,}299 & 0{,}587 & 0{,}114 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{15}$$

Una vez que el video está en escala de grises, se procesa cada cuadro con el filtro de Wiener utilizando una *vecindad* de 3×3 píxeles. Este filtro reduce parte del *ruido* original del video lo que facilita el trabajo para el siguiente paso (representado en el diagrama como “Clasificación de Bloques”). En esta etapa se aplica un algoritmo que, analizando la evolución de la media y la varianza de cada bloque, intenta decidir si las diferencias entre cuadros consecutivos se deben a movimientos del paciente (o algún objeto visible de la escena) o sólo al ruido presente en el video.

De esta forma, para cada cuadro se calcula una máscara que sólo contiene los bloques *relevantes* del mismo. A su vez, el algoritmo construye una nueva secuencia de video en la que cada cuadro difiere del anterior solo en los bloques con *información relevante* (representada en el diagrama como la flecha con la leyenda “Bloques”). El funcionamiento de este algoritmo se detalla en la sección 6.

Sobre esta nueva secuencia de video se aplica un algoritmo de compensación de movimiento basado en la *suma de diferencias absolutas* o *SAD* por sus siglas en inglés. Mediante esta técnica se busca representar la mayor porción posible de cada cuadro en base a distintas porciones del cuadro anterior. Para ello, se calcula la *SAD* para cada bloque contra una región circundante de 24×24 píxeles del cuadro anterior. Si se encuentra una región del cuadro anterior suficientemente similar a un bloque en el cuadro actual, se puede representar dicho bloque como un desplazamiento, lo que conlleva un importante ahorro de información. En la sección 7 se explica esta técnica en detalle.

Combinando los valores de la *SAD* generada durante la detección de movimiento y la máscara obtenida como resultado de la clasificación de bloques, se clasifican los bloques en base a la cantidad de información que contienen. A esta estrategia se la denomina *Clasificación por SAD* y permite variar la estrategia de compresión aplicada sobre cada bloque. También en la sección 7 se muestran en detalle los criterios de clasificación utilizados y se explican las distintas estrategias de compresión elegidas.

La sección del diagrama llamada *Reducción de entropía* agrupa un conjunto de técnicas normalmente utilizadas en métodos de compresión para reducir la

entropía de las señales a comprimir. Algunas de las técnicas que se aplican son la *transformada discreta del coseno* o *DCT*; la *cuantización*; y la *codificación diferencial*. Estas técnicas son combinadas para implementar las estrategias de compresión mencionadas anteriormente. Todo esto se explica en la sección 8.

Finalmente, en la última etapa, se toma la secuencia de video y toda la información recopilada hasta este punto y se crea una representación binaria (y comprimida de la misma). Para ser más precisos, en primer lugar se genera una secuencia de valores numéricos llamda *codificación intermedia* para luego codificarla mediante un *codificador aritmético* [5], que genera una salida binaria la cual representa el resultado del método de compresión. En la sección 8 se explica detalladamente este proceso.

6. Clasificación de bloques en relevantes o no relevantes

Como se mostró en la Sección 4, la similitud entre cuadros consecutivos no garantiza que la codificación diferencial reduzca drásticamente la entropía de la secuencia. Esto se debe a que el ruido y otros factores (como la iluminación) introducen pequeñas diferencias en los cuadros consecutivos, aún cuando a simple vista estos parecen idénticos. Para maximizar la tasa de compresión, es necesario poder distinguir cuáles de estas diferencias se deben al ruido y cuáles a movimientos reales del paciente.

Para poner nombre a las cosas, las secuencias muestran escenas y cada escena puede dividirse en dos componentes: el *plano principal* y el *fondo* (la Figura 8 ejemplifica esto sobre un cuadro de la secuencia *P1_EXT_K1 - Fragmento 1*).

Figura 8: Izquierda, el cuadro 600 de la secuencia *P1_EXT_K1 - Fragmento 1* en el que se aprecia la escena completa, en medio el *plano principal* de la escena y a la derecha el *fondo*.

El *plano principal* está compuesto por el paciente y, ocasionalmente, por algunos objetos que entran en escena. Por ejemplo, durante la mayor parte del fragmento *P1_EXT_K1 - Fragmento 1* el *plano principal* está constituido solo por el paciente, pero alrededor del cuadro 870, alguien le muestra al paciente un cuaderno; en este punto el cuaderno pasa a ser parte del plano de la escena.

Por otra parte, el *fondo* es todo lo que no cambia notoriamente de un cuadro a otro: las paredes, la cama, los equipos de monitoreo, etc. El aspecto fundamental es que el *observador* fija su atención el *plano principal* y nunca en el *fondo* de la escena.

Esto quiere decir que para reducir la cantidad de información que se busca codificar, se puede descartar tanta información del *fondo* como sea posible sin que esto afecte la utilidad del video. Para ello, a continuación se presenta un algoritmo que se basa en las propiedades estadísticas del video, llamado *algoritmo de selección de bloques para enviar*.

6.1. Estrategia del Algoritmo

El algoritmo (cuyo pseudocódigo se presenta en el Apéndice A) hace una primera clasificación de bloques en *relevantes* y *no relevantes*. Se codifican únicamente los bloques relevantes, los demás no se codifican ni se transmiten. El mismo, opera analizando cada cuadro del video bloque a bloque para determinar si contienen información correspondiente al *fondo* o al *plano principal*. los bloques que sólo contienen información perteneciente al *fondo* de la escena son llamado bloques *no relevantes*; en cambio, si al menos parte del un bloque corresponde con el *plano principal*, el mismo es considerado un bloque *relevante*. Este proceso se denomina *clasificar* bloques.

Un bloque se denomina *relevante* si se detectan cambios con respecto al cuadro anterior. Si la media de un bloque es aproximadamente igual a la media acumulada de los bloques anteriores, y la varianza del bloque es aproximadamente igual a la varianza del bloque anterior, entonces se considera que el bloque es similar a los bloques anteriores, y es declarado *no relevante*. La media acumulada se mide con respecto a la última vez en que el bloque tuvo cambios.

Luego se reconstruye cada cuadro manteniendo sin cambios (con respecto al cuadro anterior) los bloques *no relevantes* y sólo se incluyen los bloques *relevantes* del cuadro procesado. Al repetir esta operación consecutivamente sobre todos los cuadros, se obtiene como resultado una secuencia de video en la que cada cuadro está compuesto por un "collage" de bloques provenientes de distintos cuadros.

En la figura 9 se muestra un ejemplo de este proceso sobre cuatro cuadros. La fila superior muestra cada cuadro con los bloques catalogados como *relevantes* indicados con una cruz. La fila inferior muestra la salida del algoritmo cuadro a cuadro, donde cada bloque está representado por el número de su cuadro de origen.

La clasificación de bloques se basa en el análisis de la media y el desvío estándar de las diferencias entre cuadros consecutivos. La estrategia consiste en analizar la media y el desvío estándar de cada bloque en forma independiente a lo largo de la secuencia y utilizar dos umbrales para definir si el mismo es *relevante* o no.

El análisis del desvío estándar se realiza sobre la diferencia de cuadros consecutivos. Por ejemplo, se dirá que el bloque b del cuadro r es *relevante* si

$$\sigma\left(S_b^{[r-1]} - S_b^{[r]}\right) > h_\sigma \tag{16}$$

donde h_σ es el umbral definido para el desvío estándar. Esto permite detectar cambios rápidos en el bloque, producidos generalmente por movimientos en la escena y cambios de cámara.

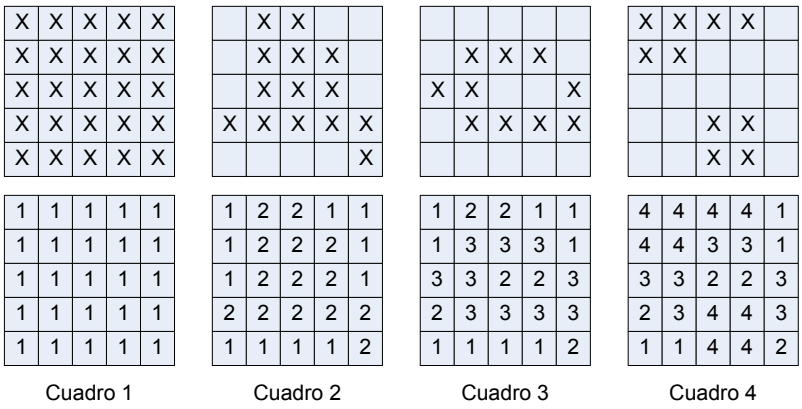


Figura 9: Ejemplo de del *algoritmo de selección de bloques para enviar* aplicado a una secuencia de cuatro cuadros. Arriba, con una cruz, se notan los bloques relevantes (con cambios) abajo, los valores indican el cuadro de origen de cada bloque.

El análisis realizado sobre la media resulta más complejo, pues se calcula sobre la diferencia entre el cuadro analizado y el promedio de una subsecuencia de longitud variable de cuadros anteriores. En este caso se dice que el bloque b del cuadro r es *relevante* si

$$\mu\left(\mu\left(S_b^{[k_b:r]}\right) - S_b^{[r]}\right) > h_\mu \tag{17}$$

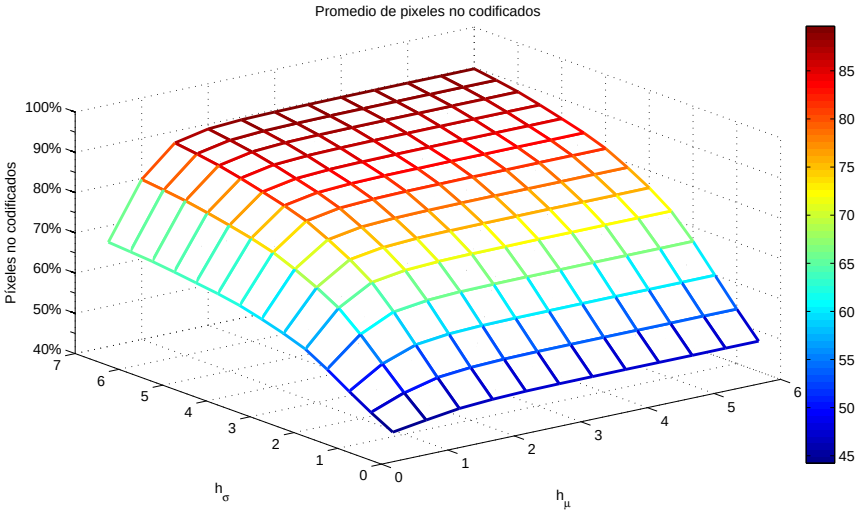
donde h_μ es el umbral definido para la media y k_b es el *cuadro clave* para el bloque b. El *cuadro clave* para un bloque dado es el índice del último cuadro en el cual dicho bloque fue catalogado como *relevante*.

El análisis sobre la media le permite al algoritmo adaptarse a cambios graduales de la escena tales como los cambios de luminosidad que pueden no ser detectados al análisis del desvío estándar.

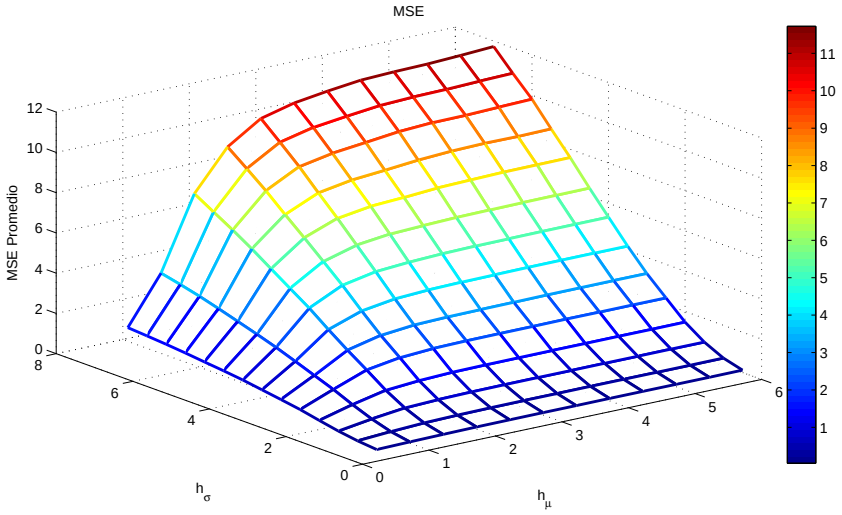
6.1.1. Ajuste de los umbrales

Para hallar los valores óptimos de los umbrales h_μ y h_σ se llevó a cabo el siguiente experimento: Utilizando el *algoritmo de selección de bloques para enviar*, se realizaron 168 corridas de prueba sobre los 5 fragmentos de video usando valores para h_μ y h_σ en pasos de 0,5 en los intervalos $[0.5, 6]$ y $[0.5, 7]$ respectivamente.

En base al resultado obtenido para cada par de valores $\left(h_\mu, h_\sigma\right)$ se calculó el porcentaje de píxeles no codificados en toda la secuencia, y el MSE de las secuencia recuperada. Promediando los valores para los 5 fragmentos se obtienen los gráficos presentados en la figura 10.



a.



b.

Figura 10: Resultados de las corridas de prueba para estimar los parámetros. (a) Porcentaje promedio de píxeles sin codificar y (b) MSE promedio, en función de los valores de los umbrales $h\mu$ y $h\sigma$.

Como se puede observar en la figura 11, para un h_μ pequeño, al incrementar el valor del umbral h_σ el MSE se incrementa casi linealmente, mientras que la cantidad de píxeles no codificados parece seguir una progresión logarítmica.

Al contrario, cuando h_σ (fijo) es grande, al aumentar h_μ el MSE crece mucho, en cambio el número de píxeles no codificado no crece tanto. Un leve aumento de píxeles no codificados implica un gran error. Esto significa que al incrementar

los valores de los umbrales la calidad del video decrece más rápido de lo que se reduce la cantidad de información.

A su vez, si los umbrales son pequeños, entonces cualquier varianza va a resultar mayor que el umbral; casi todos los cuadros van a ser catalogados como *relevantes*, y casi todos los bloques van a ser codificados; el error MSE va a ser pequeño, pero no habrá compresión.

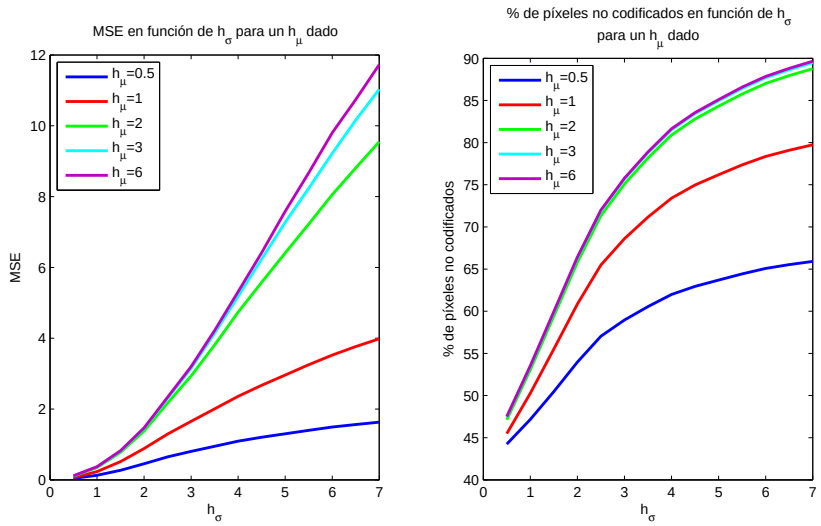


Figura 11: Análisis de la evolución del MSE y el porcentaje de píxeles no codificados en función del umbral h_σ para valores fijos de h_μ

Teniendo en cuenta esta información se seleccionaron los valores $h_\mu = 3{,}5$ y $h_\sigma = 5$ como valores de referencia para los umbrales. Estos valores producen como resultado un promedio de 81 % de píxeles sin codificar con un MSE (promedio) inferior a 10, lo que equivale a un PSNR superior a los 28,1db.

Para analizar el desempeño del *algoritmo de selección de bloques para enviar*, se calculó la entropía diferencial de las secuencias de video resultantes del mismo. Dada la forma en que este algoritmo reconstruye los cuadros de salida, al restar píxel a píxel cuadros consecutivos sucede que los bloques *no relevantes* quedan en cero mientras que sólo hay valores distintos de cero para los píxeles correspondientes a los bloques *relevantes*. De esta forma es posible calcular, a demás de la entropía diferencial, el porcentaje de bloques no codificados (que coincide con el porcentaje de bloques *no relevantes*). Estos valores se muestran en la Tabla 5

Algo que llama la atención de la Tabla 5 es que el porcentaje de bloques no codificados y el *MSE* parecen no concordar. Sería esperable que cuanto menos bloques se codifican mayor sea el *MSE*, sin embargo, esto no sucede. Para entender porqué, hay que considerar los siguientes factores:

Fragmento	MSE	Entropía Diferencial	Bloques no Codificados
P1_EXT_K1 - Fragmento 1	14,2182	2,2658	63 %
P1_EXT_K4 - Fragmento 1	6,4802	1,6373	75 %
P1_EXT_K4 - Fragmento 2	4,5869	1,1305	83 %
P2_EXT2_K2-3 - Fragmento 1	8,3615	0,3230	96 %
P2_EXT2_K2-3 - Fragmento 2	16,3287	0,9142	88 %
Promedio	9,9951	1,2542	80,94 %

Tabla 5: Valores obtenidos de aplicar el *algoritmo de selección de bloques para enviar* a los 5 fragmentos con umbrales fijos. El MSE es el promedio de errores cuadráticos sobre cada cuadro, la entropía de las diferencias se promedió sobre cada cuadro y el porcentaje de cuadros no codificados (o *no relevantes*) es sobre el total de bloques de las secuencias.

1. Los resultados corresponden a fragmentos distintos, con distintos niveles de ruido y en los cuales los pacientes presentan distintos niveles de actividad.

2. Mientras el ruido presente en un bloque genere variaciones por debajo de los umbrales, dicho bloque no será codificado.

3. Durante los periodos de reposo, se codifican muy pocos bloques, por lo tanto, en la secuencia recuperada se mantienen los mismos valores durante muchos cuadros consecutivos.

Si para todas las secuencias se codificara la misma cantidad de bloques, el valor del MSE dependería exclusivamente del nivel del ruido y el *MSE* sería mayor para las secuencias con mayor nivel de ruido. Pero al comparar videos disímiles, no es posible establecer una relación directa entre el número de bloques codificados y el MSE.

6.1.2. Umbrales Variables

Los valores para los umbrales hallados en la sección anterior se aplican a toda la secuencia por igual. Sin embargo, es posible mejorar la tasa de compresión y la calidad del video si se codifican menos cuadros durante los períodos de menor actividad y más cuando cuándo el paciente o la cámara se mueven.

Para ello se utilizan dos umbrales e_L y e_H para clasificar cada cuadro en 3 grupos: *Alta*, *Media* y *Baja* entropía diferencial, y variar así los valores de h_μ y h_σ en función de esta clasificación. La Tabla 6 muestra cómo se clasifican los cuadros en base a éstos umbrales.

Condición	Clasificación
$H_{CD}\left(S^{[c]}\right) < e_L$	*Baja*
$e_L \leq H_{CD}\left(S^{[c]}\right) < e_H$	*Media*
$e_H \leq H_{CD}\left(S^{[c]}\right)$	*Alta*

Tabla 6: Fórmulas para clasificar cuadros de una secuencia en base a la entropía diferencial respecto de los umbrales e_L y e_H. $H_{CD}\left(S^{[c]}\right)$ representa la entropía de la diferencia del cuadro C respecto de su predecesor.

Con el objetivo de hallar valores adecuados para e_L y e_H, se analizó el comportamiento de la entropía de las diferencias cuadro a cuadro. La Figura 12 muestra la entropía diferencial para los 5 fragmentos suavizada tomando promedios sobre conjuntos de 60 valores consecutivos. Las líneas punteadas muestra el valor medio par cada video.

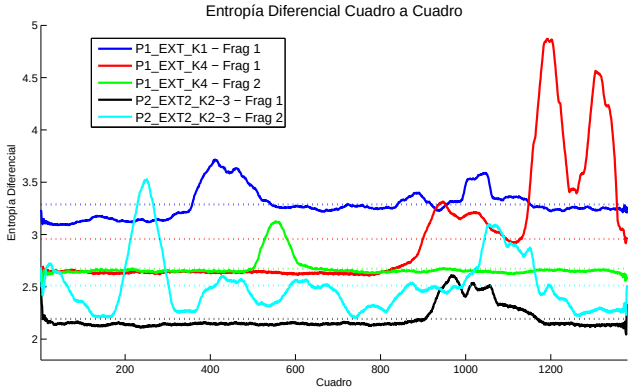


Figura 12: Entropía Diferencial cuadro a cuadro y promedio para los 5 fragmentos. Los valores fueron suavizados tomando promedios sobre conjuntos de 60 valores consecutivos

Se observó que la entropía diferencial media es distinta en todas las secuencias. Por lo tanto, de usar siempre el mismo criterio para clasificar los cuadros, se obtendrían buenas clasificaciones para algunos videos y para otros no.

Por ejemplo con los valores $e_L = 2{,}9$ y $e_H = 3{,}4$ se obtendrían buenos resultados para las secuencias *P2_EXT2_K2-3 - Fragmento 1*, *P1_EXT_K4 - Fragmento 2* y *P2_EXT2_K2-3 - Fragmento 2*. Pero en *P2_EXT2_K2-3 - Fragmento 1* todos los cuadros serían clasificados como de *Baja* entropía diferencial y en *P1_EXT_K1 - Fragmento 1* todos serían clasificados como *Media* o *Alta*, sin que ningún cuadro sea clasificado come de *Baja* entropía diferencial (aún durante los períodos de reposo).

Para evitar esto, se decidió que los umbrales e_L y e_H debían ajustarse para cada secuencia en base a la media de la entropía diferencial. Sin embargo, este enfoque presenta un inconveniente: para conocer dicha media se debe realizar un pre-procesamiento sobre el video completo antes realizar la codificación. La solución a esto es ajustar e_L y e_H cuadro a cuadro calculando la media hasta el cuadro actual.

Entonces, se definió la función

$$\mu_{H_{CD}}\left(S^{[k]}\right) = \sum_{c=0}^{k} \frac{H_{CD}\left(S^{[c]}\right)}{k} \tag{18}$$

que calcula la media de la entropía diferencial para los primeros k cuadros de la secuencia S, con la cual se ajustan los umbrales e_L y e_H para cada cuadro de la siguiente manera:

$$e_L^{[k]} = \mu_{H_{CD}}\left(S^{[k]}\right) + c_1 \tag{19}$$

$$e_H^{[k]} = \mu_{H_{CD}}\left(S^{[k]}\right) + c_2 \tag{20}$$

con $c_1 < c_2$. Las constantes c_1 y c_2 definen la distancia entre e_L y e_H y la distancia entre ambos y la media de la entropía diferencial.

Tanto a los valores de las constantes c_1 y c_2, como a los distintos valores que toman h_μ y h_σ en base a la clasificación de cada cuadro, fueron definidos empíricamente en base a pruebas realizadas sobre los 5 fragmentos de video descriptos en la Sección 4.1 y pueden no resultar óptimos para toda secuencia de video.

Para las constantes c_1 y c_2 se eligieron los valores

$$c_1 = 0 \quad y \quad c_2 = 0{,}5 \tag{21}$$

y para los umbrales h_μ y h_σ, los valores que se muestran en la Tabla 7.

Entropía diferencial del cuadro	Umbrales	
baja	$h_\mu = 5$	$h_\sigma = 6$
media	$h_\mu = 3{,}5$	$h_\sigma = 5$
alta	$h_\mu = 1{,}75$	$h_\sigma = 2{,}5$

Tabla 7: Valores para los umbrales h_μ y h_σ en función de la clasificación de los cuadros en base a su entropía diferencial.

La Figura 13 ayuda a entender cómo funciona la clasificación de cuadros. Esta figura muestra con líneas sólidas la evolución de los umbrales e_L y e_H a lo largo de la secuencia *P2_EXT2_K2-3 - Fragmento 2* (en base al criterio presentado en

la Tabla 6) y con puntos rojos, azules y verdes la clasificación de cada cuadro en base a su entropía diferencial. Como se puede ver, al inicio del video los valores de los umbrales oscilan, acompañando a la entropía diferencial de los cuadros, para luego estabilizarse.

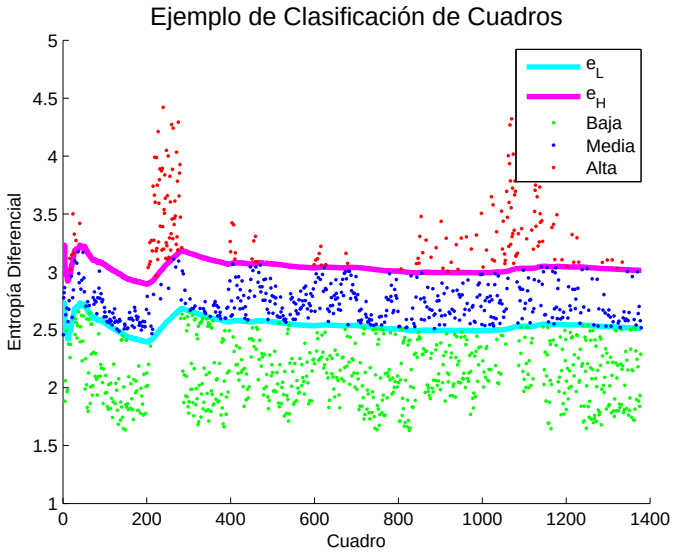


Figura 13: Ejemplo de clasificación de cuadros en base a su entropía diferencial. En líneas sólidas se muestran los umbrales

La Figura 14 muestra la clasificación de cuadros para los fragmentos de video restantes y la Tabla 8, los resultados obtenidos de aplicar el *algoritmo de selección de bloques para enviar* con umbrales variables sobre los 5 fragmentos. En relación a la variante con umbrales fijos, se observa una reducción del *MSE* de alrededor del 33 % manteniéndose la entropía y la cantidad de bloques no codificados virtualmente sin cambios (comparar con la Tabla 5). Esto se traduce en una notable mejora de la calidad de la imagen, pues el PSNR obtenido con este método es de 31,52*db*.

6.2. Reducción de ruido y resultados

Hasta este punto se ha trabajado en reducir la información redundante aplicando el *algoritmo de selección de bloques para enviar* a las secuencias sin ningún tipo de pre-procesamiento (a excepción de haberlas convertido a escala de grises). Si bien este método logra reducir la entropía diferencial en más de un 53 %, no reduce el *ruido* presente en todos los videos analizados. La reducción de ruido es importante pues mejora el rendimiento de la técnica de *compensación de movimiento* que se presenta en la Sección 7.

Fragmento	MSE	Entropía Diferencial	Bloques No Codificados
P1_EXT_K1 - Fragmento 1	9,5157	2,2114	64 %
P1_EXT_K4 - Fragmento 1	3,7611	1,6831	71 %
P1_EXT_K4 - Fragmento 2	4,6766	1,0510	84 %
P2_EXT2_K2-3 - Fragmento 1	5,7207	0,3546	95 %
P2_EXT2_K2-3 - Fragmento 2	10,1919	0,9543	87 %
Promedio	6,7732	1,2509	80,24 %

Tabla 8: Valores obtenidos de aplicar el Algoritmo 1 con umbrales variables a los 5 fragmentos. El MSE es el promedio de errores cuadráticos sobre cada cuadro, la entropía de las diferencias se promedió sobre cada cuadro y el porcentaje de cuadros no codificados es sobre el total de bloques de las secuencias.

El enfoque para disminuir el nivel de ruido se basa en aplicar el Filtro de Wiener (utilizando una vecindad de 3×3 píxeles) sobre cada cuadro de las secuencias antes de procesarlas con el *algoritmo de selección de bloques para enviar*. De este modo se consigue bajar drásticamente la entropía y aumentar la cantidad de bloques no codificados. Los resultados se muestran en la Tabla 9). Al observar los videos resultantes se descubrió que la calidad visual había caído por debajo de lo aceptable pues en muchos casos los rostros y los brazos de los pacientes se volvían indistinguibles (ver la parte derecha de la Figura 15).

Fragmento	MSE	Entropía Diferencial	Bloques No Codificados
P1_EXT_K1 - Fragmento 1	39,8699	1,2098	83 %
P1_EXT_K4 - Fragmento 1	13,3884	0,8949	86 %
P1_EXT_K4 - Fragmento 2	10,9336	0,2704	96 %
P2_EXT2_K2-3 - Fragmento 1	23,6180	0,1229	98 %
P2_EXT2_K2-3 - Fragmento 2	30,1978	0,6086	93 %
Promedio	23,6015	0,6213	91 %

Tabla 9: Resultados de aplicar el *algoritmo de selección de bloques para enviar* con los umbrales originales sobre los fragmentos previamente procesados con el filtro de Wiener. El MSE es el promedio de errores cuadráticos sobre cada cuadro, la entropía de las diferencias se promedió sobre cada cuadro y el porcentaje de cuadros no codificados es sobre el total de bloques de las secuencias.

Unos de los efectos de aplicar el filtro de Wiener fue que, además de reducir el *ruido*, cambió sutilmente algunas características de los videos, haciendo que los bordes sean más suaves y algo difusos. Estos cambios no son lo suficientemente

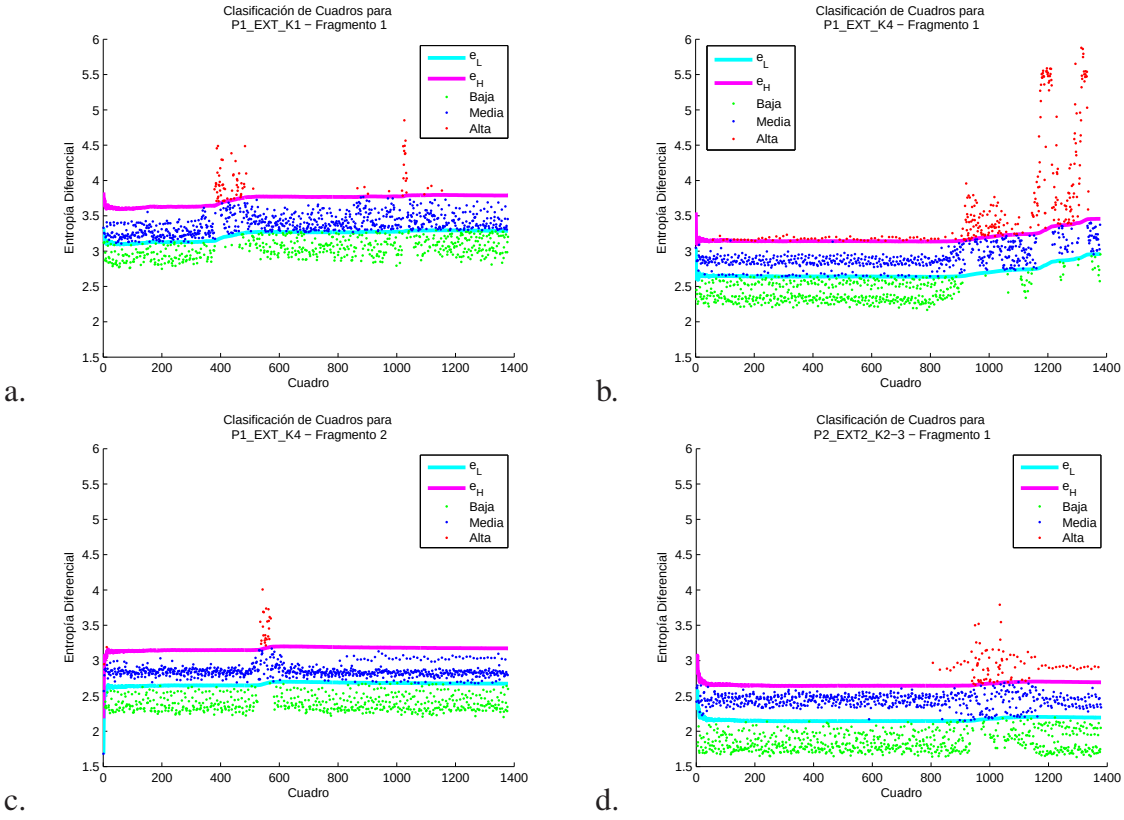


Figura 14: Clasificación de los cuadros en base a la entropía diferencial para las secuencias (a) *P1_EXT_K4 - Fragmento 1*, (b) *P1_EXT_K4 - Fragmento 2*, (c) *P2_EXT2_K2-3 - Fragmento 1* y (d) *P2_EXT2_K2-3 - Fragmento 2*. Los umbrales se muestran como líneas sólidas.

grandes para ser notados simple vista, pero sí para hacer que el *algoritmo de selección de bloques para enviar* no pueda distinguir entre el *fondo* y el *plano principal* de la escena cuando el paciente realizaba movimientos lentos.

Como consecuencia, fue necesario buscar nuevos valores para los umbrales h_μ y h_σ. Luego de realizar múltiples corridas de prueba se hallaron los valores que se muestran en la Tabla 10. La Figura 15 permite comparar el cuadro 190 de la secuencia *P2_EXT2_K2-3 - Fragmento 2* procesado con los umbrales originales (que se encuentran en la Tabla 7) y los nuevos.

Otra consecuencia de utilizar el Filtro de Wiener es un notable aumento del *MSE*. Esto se debe, en parte, a que el filtro de Wiener está diseñado para disminuir el ruido en (i) señales estáticas cuyas propiedades estadísticas no cambian con el tiempo ni con el lugar, (ii) para situaciones en que la señal y su ruido no tienen correlación. Ninguno de los dos supuestos se cumple para los videos analizados

Entropía del cuadro	Umbrales	
baja	$h_\mu = 5$	$h_\sigma = 2$
media	$h_\mu = 3{,}5$	$h_\sigma = 2$
alta	$h_\mu = 1{,}75$	$h_\sigma = 1$

Tabla 10: Valores variables para los umbrales h_μ y h_σ corregidos para trabajar sobre secuencias procesadas con el Filtro de Wiener.

Figura 15: Cuadro 190 de la secuencia *P2_EXT2_K2-3 - Fragmento 2* filtrada con Wiener utilizando los umbrales originales (izquierda) y con los umbrales corregidos para el Filtro de Wiener (derecha).

en este trabajo, sin embargo, el uso del filtro mejora notablemente el rendimiento del *algoritmo de selección de bloques para enviar* y las consecuencias visuales son mínimas.

A su vez, se debe considerar que el MSE es una fórmula para medir la calidad del video procesado, que no siempre refleja nuestra percepción del error. En este trabajo se usa, además, una noción más subjetiva de la calidad: Si el paciente se puede ver con claridad y los especialistas puede estudiar sus expresiones y movimientos durante un ataque entonces el resultado es de buena calidad. A lo largo de este trabajo se utiliza el *MSE* sólo como una referencia mientras que a la calidad de los resultados se mide en forma subjetiva analizando visualmente los videos.

La Tabla 11 muestra los resultados de procesar los 5 fragmentos filtrados con Wiener utilizando los nuevos umbrales. Comparando estos resultados con la Tabla 8 se advierte que la cantidad de bloques no codificados es similar, la entropía diferencial se redujo un 17 % y, como se mencionó anteriormente, el MSE aumentó a más del doble.

Fragmento	MSE	Entropía Diferencial	Bloques No Codificados
P1_EXT_K1 - Fragmento 1	24,0344	1,8723	66 %
P1_EXT_K4 - Fragmento 1	7,0761	1,4273	73 %
P1_EXT_K4 - Fragmento 2	7,6860	0,8141	85 %
P2_EXT2_K2-3 - Fragmento 1	12,4551	0,3337	95 %
P2_EXT2_K2-3 - Fragmento 2	16,8791	0,9065	86 %
Promedio	13,6261	1,0708	81 %

Tabla 11: Resultados de aplicar el *algoritmo de selección de bloques para enviar* sobre los fragmentos previamente procesados con el filtro de Wiener usando los umbrales corregidos. El MSE es el promedio de errores cuadráticos sobre cada cuadro, la entropía de las diferencias se promedió sobre cada cuadro y el porcentaje de cuadros no codificados es sobre el total de bloques de las secuencias.

6.3. Distorsiones

En la sección anterior se vio que debido a los cambios en las propiedades de las secuencias producidos como consecuencia de aplicar el *filtro de Wiener* comenzaron a aparecer distorsiones en las imágenes que causaron una notable pérdida de calidad. En esta sección se analizan dichas distorsiones.

Eliminar información de una secuencia de video no es gratuito, especialmente si se hace de forma tan "agresiva" como lo hace el *algoritmo de selección de bloques para enviar*. Como consecuencia los videos pueden presentar dos tipos de distorsiones bien definidas que se acentúan cuando los valores elegidos para los umbrales h_μ y h_σ son altos. En la Figura 16 se observa el cuadro 40 de la secuencia *P2_EXT2_K2-3 - Fragmento 2* y el mismo cuadro para la secuencia procesada utilizando los umbrales ($h_\mu = 3{,}5, h_\sigma = 5$) y ($h_\mu = 10, h_\sigma = 7$).

La distorsión más notoria es un conjunto de cuadrados que producen cierto efecto de pixelado. Estos cuadrados son, en realidad bloques que, debido a su similitud en media y varianza, han sido catalogados como *no relevantes* durante demasiados cuadros consecutivos, aún cuando la imagen ha cambiado mucho. Esto sucede principalmente sobre el fondo de la escena (o en superficies grandes y monótonas) cuando ocurren cambios de luminosidad en la escena. Estas distorsiones son llamadas *distorsiones de tipo 1*.

Un segundo efecto secundario de este algoritmo es la aparición de pequeños puntos o manchas que claramente no coinciden con el cuadro original y son fragmentos de bordes de objetos en movimiento que son considerados por el algoritmo como parte del *fondo* de la escena. Para ejemplificar esto se puede analizar la siguiente situación: Supongamos que la mano del paciente se encuentra sobre

las sábanas blancas y que al dividir la escena en bloques un pequeña porción de un dedo ocupa no más que un par de píxeles de un bloque que muestra principalmente la sábana. Si el paciente mueve el brazo de modo que el dedo deje de estar en dicho bloque, es muy probable que la variación producida en el mismo no sea lo suficientemente alta para que el algoritmo detecte un cambio y de esa forma el bloque podría seguir siendo catalogado como *no relevante* durante varios cuadros. Este ejemplo se ilustra en la Figura 17. A estas distorsiones se las llama *distorsiones de tipo 2*.

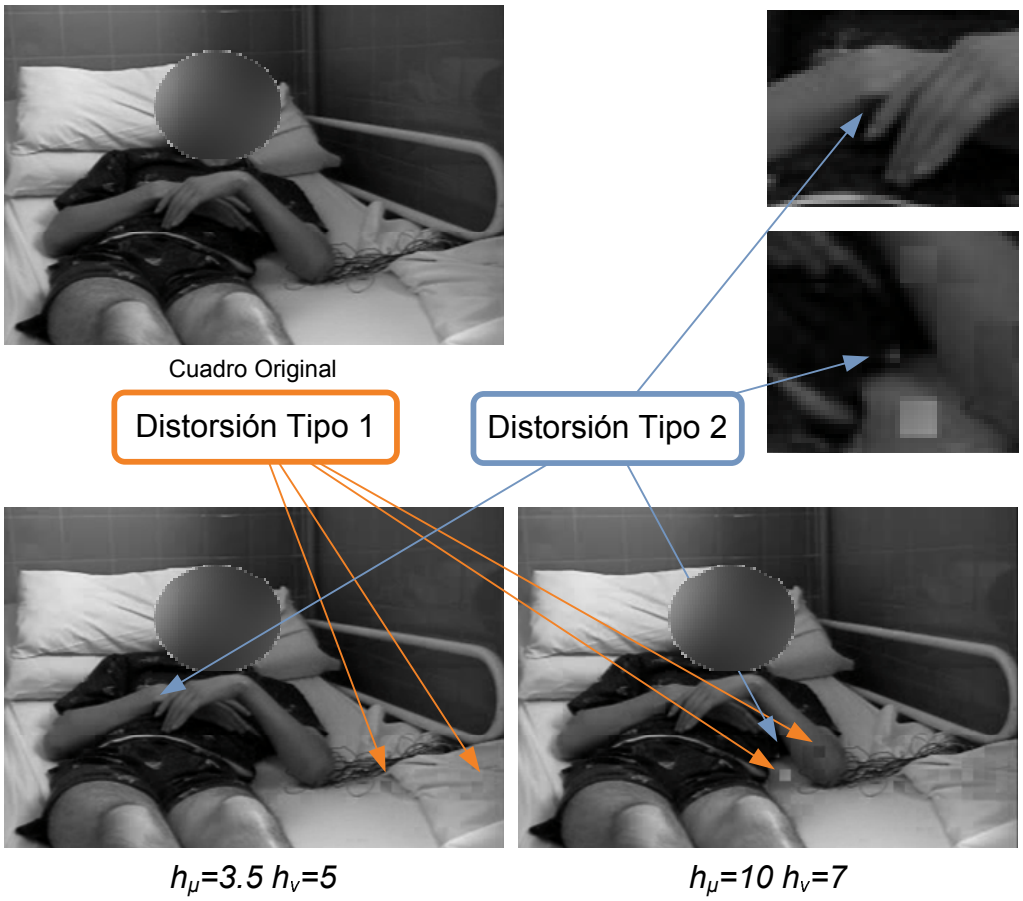


Figura 16: Distorsiones producidos por el *algoritmo de selección de bloques para enviar*. *Arriba a la izquierda*: el cuadro 40 de la secuencia *P2_EXT2_K2-3 - Fragmento 2*. *Abajo*: el mismo cuadro obtenido de procesar la secuencia con dos conjuntos distintos de valores para los umbrales h_μ y h_σ. *Arriba a la derecha*: ampliaciones en la que se pueden observar las distorsiones con más detalle.

La existencia de este tipo de distorsiones son consecuencia directa de la es-

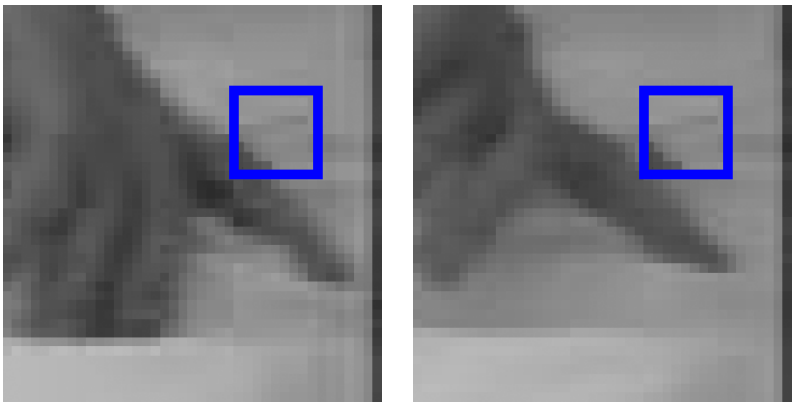


Figura 17: Ejemplo distorsión de tipo 2 sobre la mano del paciente. El recuadro azul delimita un bloque de 8×8 píxeles. El cuadro de la izquierda precede al de la derecha. Cuando el paciente mueve la mano, el bloque no cambia y una pequeña porción del dedo se sigue mostrando en su posición anterior. Esto da lugar a un punto obscuro que no pertenece al video original.

trategia usada por el *algoritmo de selección de bloques para enviar*, sin embargo su impacto en la calidad del video es nulo en tanto no se encuentren en las partes del cuerpo del paciente que son objeto de estudio para los especialistas (ojos, boca, brazos, etc.).

7. Compensación de movimiento

Con la clasificación en *relevantes* o *no relevantes*, y la posterior codificación en la cual únicamente se envían los bloques relevantes, se elimina gran parte de la información redundante de las secuencias de video. Sin embargo aún se puede mejorar la compresión aplicando la técnica de *compensación de movimiento* [18]. Esta consiste en buscar en el cuadro anterior reconstruido un bloque similar al bloque que se está por enviar. Si el bloque hallado es casi idéntico al bloque actual, solamente se envía la referencia a la ubicación del bloque similar encontrado. Si el bloque hallado tiene algunas similitudes con el bloque actual, se codifican las diferencias. Y si no tiene similitudes con el bloque actual, se lo codifica como una imagen (modo intra coding).

Para dar un ejemplo, supongamos que se toman dos cuadros consecutivos k y $k+1$ de una secuencia T, en el primero el bloque b muestra parte de un dedo del paciente y a su vez, en el cuadro siguiente el mismo dedo se encuentra en el bloque c. Para representar estos cambios, el *algoritmo de selección de bloques para enviar* debe codificar el bloque b del cuadro k y los bloques b y c del cuadro $k+1$ (el c porque es donde está ahora el dedo y el b porque ahora contiene otra cosa). Sin embargo, es muy probable que los bloques $T_b^{[k]}$ y $T_c^{[k+1]}$ sean muy similares o incluso iguales. Si este fuera el caso, en lugar de codificar 3 bloques se podrían codificar solo los bloques $T_b^{[k]}$ y $T_b^{[k+1]}$ y en lugar del bloque $T_c^{[k+1]}$ podríamos codificar las coordenadas del bloque $T_b^{[k]}$. Sin embargo, como el decodificador no tiene información del bloque $T_b^{[k]}$, sino una aproximación del mismo (el cuadro anterior reconstruido) la búsqueda del bloque b se hace en el cuadro k reconstruido. Esta técnica es ampliamente utilizada en los estándares [2].

Aplicando la técnica de compensación de movimiento a toda la secuencia es posible reducir notablemente la cantidad de información que debe ser codificada y así mejorar el rendimiento del método de compresión.

7.1. Detección de similitudes y segunda clasificación de los bloques

Se trata de una segunda clasificación, que va a determinar de qué manera se codifican los bloques. La estrategia de *compensación de movimiento* utilizada en el presente trabajo se basa en buscar similitudes para cada uno de los bloques *relevantes* de cada cuadro en una región de 24×24 píxeles del cuadro anterior llamada *superbloque*. Para cada bloque b dentro de un cuadro k reconstruido se define un *superbloque* $\widehat{B}$ en $k-1$ tal que contenga al bloque b justo en su centro. Si el bloque b está en el borde del cuadro, simplemente se rellenan con ceros los puntos de $\widehat{B}$ que quedan fuera del cuadro.

Llamando $B = T_b^{[k]}$ al bloque actual de 8×8, que comienza en las coordenadas x_b, y_b. Se busca en el cuadro anterior reconstruido un bloque $\hat{B} = \hat{T}_c^{[k-1]}$ de 8×8, que comience en las coordenadas x_c, y_c, donde $x_c = x_b + \Delta x$, $y_c = y_b + \Delta y$, con $-8 \leq \Delta x \leq 8$ y $-8 \leq \Delta y \leq 8$ de manera que $SAD(B, \hat{B})$ sea mínimo.

En base al valor $SAD(B, \hat{B})$ se puede estimar que tan similares son el bloque B y su contraparte $\hat{B}$ hallada en el cuadro anterior, y así elegir cómo codificarlo. En este trabajo se contemplan tres modos de codificación y utiliza dos umbrales h_L y h_H para determinar cual usar. Estos dos umbrales sumados al resultado de clasificación de bloques en relevantes y no relevantes, permite clasificar cada bloque en cuatro clases (0, 1, 2, y 3) tal como se muestra en la Tabla 12.

Clase	Relevancia	Mínimo $SAD(B, \hat{B})$	Representación
0	*no relevante*	-	Nada
1	*relevante*	$SAD(B, \hat{B}) \leq h_L$	$(\Delta x, \Delta y)$
2	*relevante*	$h_L < SAD(B, \hat{B}) \leq h_H$	$\left(\Delta x, \Delta y, B - \hat{B}\right)$
3	*relevante*	$h_h < SAD(B, \hat{B})$	$(\Delta x, \Delta y, B)$

Tabla 12: Segunda clasificación de los bloques.

Según su clasificación, la información necesaria para codificar un bloque es distinta y, por lo tanto, lo es su representación:

- Los bloques de *Clase 0* han sido catalogados como *no relevantes* por el *algoritmo de selección de bloques para enviar* y no es necesario codificarlos.
- Los bloques de *Clase 1* son virtualmente idénticos a alguna región del cuadro anterior, por lo tanto se los representa como un par de valores de desplazamiento.
- Si un bloque fue catalogado como de *Clase 2*, significa que el algoritmo ha encontrado una región similar en el cuadro anterior, pero la misma no puede ser utilizada tal cual es. Para representar estos bloques necesitaremos un par de valores de desplazamiento más una matriz de 8×8 conteniendo las diferencias.
- Por último, los bloques de *Clase 3* no pueden expresarse a partir del cuadro anterior y deben ser codificados completos. Estos bloques son representados por una matriz de 8×8.

Los valores de los umbrales h_L y h_H definen la *sensibilidad* del método y afectan directamente a la tasa de compresión que puede conseguirse y a la calidad

del video resultante. El umbral h_L controla que tan similar al bloque codificado debe ser una región del cuadro anterior para poder reemplazarlo. Por otro lado, el umbral h_H define cuándo se deberá transmitir un bloque completo porque no puede ser recuperado del cuadro anterior.

Si el valor de h_L es muy alto se obtiene una alta tasa de compresión, pues muchos bloques serán reemplazados por coordenadas de desplazamiento (*Clase 1*), pero el resultado será de muy mala calidad ya que los cuadros terminarán siendo un *mosaico* de fragmentos de cuadros anteriores con muchas diferencias respecto del cuadro original.

En cambio, si se escoge un valor demasiado alto para h_H habrá gran cantidad de bloques de *Clase 2*. La calidad del video resultante no será tan afectada, pero sí lo será la tasa de compresión: para codificar un bloque de *Clase 2* se incluye la diferencia entre el bloque original y la región seleccionada del cuadro anterior, si ambas cosas son muy distintas entre sí, las diferencias codificadas contendrán tanta información como el bloque completo o incluso más.

En este trabajo se definieron los valores de h_L y h_H según el siguiente criterio:

Si la *SAD* entre el bloque a codificar y la región más similar hallada en el cuadro anterior es menor o igual a 255, el bloque será considerado de *Clase 1*. Esto significa que ambos bloques pueden diferir totalmente en a los sumo un pixel. Claro está que esto último es un caso extremo pues lo más probable es que todos los píxeles tengan pequeñas diferencias, pero en cualquier caso la suma absoluta de las diferencias no debe superar el valor:

$$h_L = 255 \tag{22}$$

Con respecto a h_H, para considerar a un bloque de *Clase 2* la tolerancia es de a lo sumo 4 píxeles totalmente diferentes, de modo que se definió:

$$h_H = 1024 \tag{23}$$

7.2. Resultados

Los resultados de aplicar la técnica de *compensación de movimiento* a la salida del *algoritmo de selección de bloques para enviar* se muestran en la Tabla 13. Comparando estos números con la Tabla 11 se observa que la entropía se redujo casi 4 veces, mientras que el MSE aumentó un 48 %.

La Tabla 14 muestra el porcentaje de bloques pertenecientes a cada clase. Los bloque de *Clase 0* son aquellos catalogados como *no relevantes*, de modo que se toman en cuenta sólo los bloques de *clase* 1, 2 y 3. La Tabla 15 muestra cuántos bloques *relevantes* son asignados a cada clase.

Si se considera la cantidad de valores necesarios para codificar los bloques de cada clase, tal como se muestra en la Tabla 16, se puede calcular la cantidad

de valores necesarios para codificar cada secuencia antes y después de aplicar la técnica de *compensación de movimiento* y de esta forma calcular cuánta información se evita codificar gracias a esta técnica. Estos valores se muestran en la Tabla 17.

Fragmento	MSE	Entropía
P1_EXT_K1 - Fragmento 1	33,3277	0,6135
P1_EXT_K4 - Fragmento 1	12,9324	0,2843
P1_EXT_K4 - Fragmento 2	13,2514	0,0889
P2_EXT2_K2-3 - Fragmento 1	16,9797	0,0694
P2_EXT2_K2-3 - Fragmento 2	24,5837	0,3574
Promedio	20,2150	0,2738

Tabla 13: Resultados obtenidos al aplicar la técnica de *compensación de movimiento* y la segunda selección de bloques sobre la salida del *algoritmo de selección de bloques para enviar* a las secuencias procesadas con el filtro de Wiener. El MSE es el promedio de errores cuadráticos sobre cada cuadro. La entropía se calculó como el promedio, por cada cuadro, de la información necesaria para representar cada bloque en base al resultado de la segunda clasificación de bloques (incluyendo un mapa de bits indicando la clase de cada bloque), sin aplicar ningún tipo de transformación ni cuantización.

	Clase 0	*Clase 1*	*Clase 2*	*Clase 3*
P1_EXT_K1 - Fragmento 1	64,59 %	20,53 %	14,72 %	0,16 %
P1_EXT_K4 - Fragmento 1	71,73 %	23,73 %	4,39 %	0,15 %
P1_EXT_K4 - Fragmento 2	84,01 %	14,10 %	1,84 %	0,05 %
P2_EXT2_K2-3 - Fragmento 1	92,89 %	6,05 %	0,99 %	0,07 %
P2_EXT2_K2-3 - Fragmento 2	84,73 %	9,50 %	5,59 %	0,18 %
Promedio	79,59 %	14,78 %	5,51 %	0, 12 %

Tabla 14: Porcentaje de bloques asignados a cada clase para cada secuencia de video sobre un total de 1.898.160 bloques por video.

	Clase 1	*Clase 2*	*Clase 3*
P1_EXT_K1 - Fragmento 1	57,98 %	41,57 %	0,45 %
P1_EXT_K4 - Fragmento 1	83,94 %	15,53 %	0,53 %
P1_EXT_K4 - Fragmento 2	88,18 %	11,51 %	0,31 %
P2_EXT2_K2-3 - Fragmento 1	85,09 %	13,92 %	0,98 %
P2_EXT2_K2-3 - Fragmento 2	62,21 %	36,61 %	1,18 %
Promedio	75,48 %	23,83 %	0, 69 %

Tabla 15: Porcentaje de bloques por clase sobre los bloques *relevantes* de cada secuencia.

Clase	M	Cantidad de Valores
Clase 0	0	No es necesario codificar información para estos bloques. Al momento de decodificar la secuencia se repite el bloque del cuadro anterior.
Clase 1	2	Solo se necesita codificar las coordenadas de desplazamiento horizontal y vertical.
Clase 2	66	Se codifican las coordenadas de desplazamiento (2 valores) más un bloque de diferencias que consiste de 64 valores.
Clase 3	64	No se puede aplicar *compensación de movimiento* sobre estos bloques, por lo tanto se los debe codificar completos.

Tabla 16: Segunda clasificación de bloques. M: cantidad de valores que deben codificarse para cada clase de bloque.

Fragmento	Cantidad de Valores		Porcentaje Codificado
	Antes	Después	
P1_EXT_K1 - Fragmento 1	41.464.704	8.045.070	19 %
P1_EXT_K4 - Fragmento 1	32.605.504	3.420.928	10 %
P1_EXT_K4 - Fragmento 2	17.429.376	843.994	5 %
P2_EXT2_K2-3 - Fragmento 1	6.380.864	704.828	11 %
P2_EXT2_K2-3 - Fragmento 2	16.501.184	4.022.764	24 %
Promedio	22.876.326	3.407.516	14 %

Tabla 17: Por cada fragmento procesado con el filtro de Wiener, Antes: cantidad de valores necesarios para codificar la información resultante al realizar la clasificación de bloques *relevantes* y *no relevantes*; Después: cantidad de valores necesarios para codificar la información resultante de la segunda selección de bloques.

8. Cuantización y Codificación

En la sección anterior se detalló el mecanismo por el cual cada bloque puede ser clasificado dentro de cuatro clases y se mencionó qué información se debe codificar en cada caso (ver Tabla 12). En esta sección se explica cómo, a partir de dicha clasificación y de las salidas del *algoritmo de selección de bloques para enviar*, se codifica una secuencia de video en formato binario.

Para obtener la representación binaria se siguen los siguientes pasos: en primer lugar se *transforma* y *cuantiza* la información de los bloques de *Clase 2* y *Clase 3* siguiendo los mismos mecanismos que en el estándar JPEG [21], esto permite mejorar la tasa de compresión; luego se genera una representación intermedia en la cual el video es representado como una secuencia de símbolos de un alfabeto finito buscando reducir la cantidad de ceros que deben ser codificados; finalmente, se aplica un codificador aritmético a dicha secuencia para generar la representación binaria del video comprimido.

A continuación se detalla cada uno de estos pasos.

8.1. Transformación y Cuantización de matrices

Las representaciones de los bloques de *Clase 2* y *Clase 3* incluyen matrices de 8×8 valores. En el primer caso dicha matriz contiene las diferencias entre el bloque codificado y alguna región del cuadro anterior y en el segundo caso la matriz es el bloque en si mismo.

El objetivo es reducir al mínimo posible la entropía de estas matrices mediante el uso de la cuantización, minimizando el impacto visual que esto genera. Para ello se transforman las matrices mediante la *transformada discreta del coseno* para luego cuantizar sus transformadas.

La *DCT* tiene una buena capacidad de compactación de la energía al dominio transformado, es decir, que la *transformada discreta del coseno* consigue concentrar la mayor parte de la información en pocos coeficientes transformados, esto significa que la matriz transformada tiene unos pocos valores altos y un gran número de valores cercanos a cero, los cuales se anulan al realizar la cuantización.

8.1.1. Codificación de bloques de Clase 2

Antes de codificar los bloques de *Clase 2*, se debe transformar y cuantizar la componente $B - \hat{B}$ contenida en la representación de estos bloques (ver Tabla 12). Para ello se aplica la transformada *DCT* y luego se cuantiza el resultado en intervalos de 16 valores enteros. La fórmula matemática de esta operación aplicada a un bloque B de tamaño 8×8 de *Clase 2* sería:

$$Q_{ij} = fix\left(\frac{DCT\left(B - \hat{B}\right)_{(i,j)}}{16}\right) \tag{24}$$

con $0 \leq i < 8$ y $0 \leq j < 8$ donde $fix(x)$ presenta la parte entera de x, si x es positivo. Si es negativo, representa el entero más cercano en dirección al 0.

8.1.2. Codificación de bloques de Clase 3

La transformada *DCT* presenta un mejor comportamiento cuando el dominio es simétrico con respecto al cero, es decir, cuando todos los valores den la matriz a transformar se encuentran en un mismo intervalo $[-R, R]$. Ese es el caso para los *bloques de diferencias* pues sus valores se encuentran en el intervalo $[-255, 255]$, sin embargo, no es así para los bloques de *Clase 3*. Los valores de estos últimos se encuentran en el intervalo $[0, 255]$, por lo tanto, antes de aplicar la *DCT* se sustrae 128 a todos los valores del bloque para así trasladarlos al intervalo $[-128, 127]$.

Con respecto a la cuantización, también existe una diferencia importante entre ambas clases de bloques, pues los bloques de *Clase 3* son, básicamente, pequeños fragmentos de imágenes y por lo tanto deben ser cuantizados minimizando los efectos visuales sobre la imagen. Lo que se hace es cuantizar cada elemento de la matriz con un valor diferente, de modo tal de conservar más información correspondiente a las frecuencias bajas (predominantes en las fotografías y filmaciones) que a las frecuencias altas [21]. La Figura 18 muestra la matriz de cuantización P utilizada para este tipo de bloques.

La siguiente fórmula corresponde a la transformación y cuantización de un bloque B de tamaño 8×8 de *Clase 3* :

$$\tilde{B}_{ij} = fix\left(\frac{DCT\,(B - 128)_{(i,j)}}{P_{(i,j)}}\right) \tag{25}$$

con $0 \leq i < 8$ y $0 \leq j < 8$ donde $fix(x)$ presenta la parte entera de x (o el entero más cercano en dirección al 0, si x es negativo) y $B - 128$ representa el sustraer 128 a cada componente de B.

8.2. Codificación intermedia

Previamente a codificar en forma binaria el video comprimido mediante un codificador aritmético, es necesario representar el video como una secuencia de símbolos, correspondientes a un alfabeto finito, que incluya la clasificación de cada bloque, los vectores de desplazamiento y las matrices correspondientes a cada bloque.

$$\begin{pmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{pmatrix}$$

Figura 18: Matriz de cuantización del estándar JPEG utilizada para cuantizar los bloques de *Clase 3*

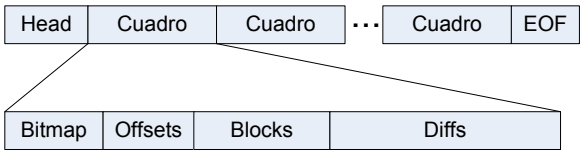


Figura 19: Esquema general de la *codificación intermedia*. Arriba se muestra el esquema para una secuencia de video completa y abajo el esquema correspondiente a un único cuadro.

En este caso, se utiliza como alfabeto a los valores enteros con signo que pueden expresarse con 8 bits, lo que equivale al intervalo $[-2^7, 2^7 - 1]$ (similar al intervalo de valores que puede tomar una variable de tipo **char** en lenguajes de programación como *Java* o *C*). Esta elección simplifica muchos aspectos de la implementación pues permite utilizar cualquier codificador aritmético implementado para secuencias de bytes.

La Figura 19 muestra el esquema general de la *codificación intermedia*. La misma consiste en un pequeño encabezado o *header* que contiene información de control necesaria para decodificar el video y luego una sucesión de cuadros.

El *header* consiste de 2 valores que representan el tamaño de los cuadros en filas y columnas. Como este método procesa el video en base a bloques de 8×8 píxeles, es posible expresar el tamaño de la imagen en base a cantidad de bloques y así utilizar sólo dos símbolos para representar tamaños de hasta 2040×2040 píxeles. De este modo, si los cuadros son de tamaño $M \times N$ los primeros dos

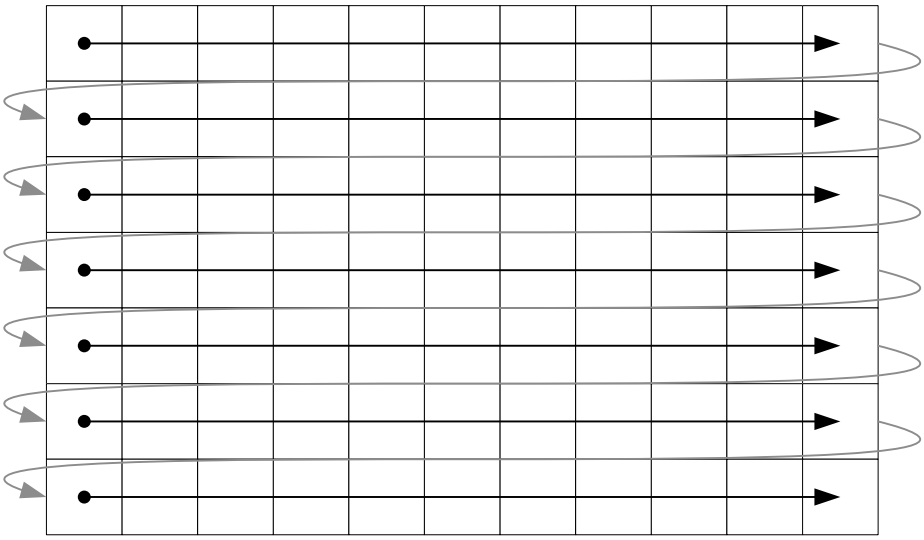


Figura 20: Representación del orden en que se codifican los bloques en las distintas secciones de la *codificación intermedia*.

valores de la codificación intermedia V quedan definidos como

$$\begin{aligned} V_0 &= \frac{M}{8} - 128 \\ V_1 &= \frac{N}{8} - 128 \end{aligned}$$

Luego del *header* siguen, uno tras otro, los cuadros que componen el video. La parte inferior de la Figura 19 muestra cómo se representan los mismos.

La información de cada cuadro se divide, en cuatro secciones consecutivas: la sección *bitmap*, conteniendo la clasificación de cada bloque; la sección *offset*, que contiene los valores de desplazamiento para los bloques de *Clase 1* y *Clase 2*; la sección *blocks*, que contiene los bloques de *Clase 3*; y, finalmente, la sección *diffs* que contiene los *bloques de diferencias* para los bloques de *Clase 2*. Dentro de cada sección la información se ordena por filas de bloques, es decir, primero los valores de la fila 0, luego los de la fila 1, y así sucesivamente tal como se muestra en la Figura 20.

La sección *bitmap* se codifica como una secuencia de $M \times N/64$ valores en el intervalo $[0, 3]$ de acuerdo a la clase. Luego, en la sección *offsets*, se codifican uno tras otro sólo los valores de desplazamiento para los bloque de *Clase 1* y *Clase 2*. Estos valores son codificados como enteros en el rango $[-8, +8]$. Por cada par de valores primero se codifica el índice de desplazamiento horizontal y luego el vertical.

Luego, en la sección *blocks*, se colocan uno tras otro los valores resultantes de transformar y cuantizar los bloques de *Clase 3*. Una característica de la transformada *DCT* es que al aplicarla a imágenes los valores altos de la transformada se agrupan en las primeras filas y columnas de la matriz. Para explotar esta propiedad, cada bloque es codificado siguiendo un *zig-zag* tal como se muestra en la Figura 21. Al secuenciar los valores de esta manera existe una alta probabilidad de que los últimos valores del bloque (muchas veces más de la mitad) sean sólo ceros. Por esto, en lugar de codificar los 64 valores, sólo se incluye hasta el último valor distinto de cero y luego se coloca un *marca de fin de bloque* o *EOB*, indicando que el resto de los valores son ceros. De darse el caso de que el último valor del bloque es distinto de cero, la marca de *EOB* es omitida.

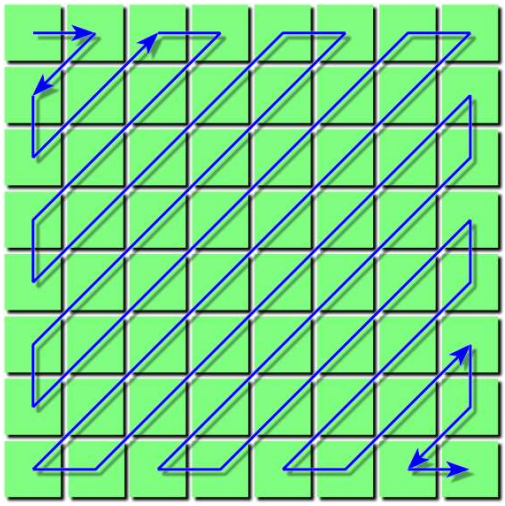


Figura 21: Representación del orden en que se codifican los valores de los bloques resultantes de aplicar la transformada DCT.

Del análisis numérico de la transformada DCT [21] para bloques de 8×8 surge que si los 64 valores son enteros de N bits, la parte entera de la transformada puede crecer en a lo sumo 3 bits. Los bloque de *Clase 3* están definidos por valores en el intervalo $[-128, 127]$ o lo que es igual $[-2^7, 2^7 - 1]$, por lo cual,al aplicar la *DCT* se obtendrán 64 valores en el intervalo $[-2^{10}, 2^{10} - 1]$. Si se toman ambos extremos de éste intervalo y se los cuantiza por el menor valor de la matriz de cuantización, se obtiene el rango de valores de un bloque de *Clase 3* transformado y cuantizado $\tilde{B}$.

$$MIN\left(\tilde{B}\right) = \quad fix\left(\frac{-2^{10}}{10}\right) = fix\left(\frac{-1024}{10}\right) = -102 \tag{26}$$

$$MAX\left(\tilde{B}\right) = fix\left(\frac{2^{10}-1}{10}\right) = \quad fix\left(\frac{1023}{10}\right) = \quad 102 \tag{27}$$

Las Ecuaciones 26 y 27 permiten concluir, en primer lugar, que es posible utilizar cualquier valor mayor a 102 o menor a -102 como marcador *EOB* (se utilizará el valor -128). En segundo lugar, que con el alfabeto elegido es posible representar cualquier bloque de *Clase 3* con no más de un símbolo por cada valor.

Finalmente se codifica la sección *diffs* conteniendo los *bloques de diferencias* ya transformados y cuantizados con la Fórmula 24. Al igual que con los bloques de *Clase 3*, los *bloques de diferencias* se secuencian en *zig-zag* y se utiliza una marca de fin de bloque para reducir la cantidad de valores necesarios.

Los *bloques de diferencias* están formados por valores en el intervalo $[-255, 255]$ o, lo que es igual, $[-2^8 + 1, 2^8 - 1]$. Del análisis los valores resultantes luego de transformar y cuantizar estos bloques, tal como se muestra en las Ecuaciones 28 y 29, surge que cada valor de los *bloques de diferencias* también puede ser codificado con un único símbolo en el alfabeto elegido. Así mismo, el valor -128 queda reservado nuevamente como marcador de fin de bloque.

$$MIN(\tilde{}) = fix\left(\frac{(-2^8+1)*2^3}{16}\right) = fix\left(\frac{-2040}{16}\right) = -127 \tag{28}$$

$$MAX\left(\tilde{B}\right) = \quad fix\left(\frac{(2^8-1)*2^3}{16}\right) = \quad fix\left(\frac{2040}{16}\right) = \quad 127 \tag{29}$$

8.3. Codificación aritmética

En la sección anterior se mostró cómo se codifican los bloques de un cuadro de video en una secuencia de valores en el intervalo $[-128, 127]$. La última etapa en este proceso de compresión consiste en codificar dicha secuencia en forma binaria, utilizando la menor cantidad de bits posibles. Para ello se utiliza un *codificador aritmético* [17].

Como se explica en [5], la *codificación aritmética* es una variante de las técnicas conocidas como *compresión estadística* o *compresión basada en la entropía*. El objetivo de estas técnicas es codificar una secuencia de entrada (o fuente) representando a cada símbolo de la misma utilizando códigos binarios de longitud variable de modo tal que la longitud media de los datos codificados sea menor que la obtenida con códigos de longitud fija. La construcción de este tipo de códigos se basa en la propiedad del prefijo, según la cual, ninguna secuencia de bits

que represente a un símbolo del alfabeto podrá aparecer como subsecuencia inicial de otra secuencia de longitud mayor que represente a otro símbolo. Con esta propiedad se asegura que estos códigos sólo admiten una única posibilidad para ser decodificados.

En particular, un *codificador aritmético* representa cada símbolo del alfabeto de entrada como un número de punto flotante. El proceso de codificación se basa en asignar a cada símbolo un intervalo entre 0 y 1, de forma que la amplitud de cada intervalo sea igual a la probabilidad que tiene dicho símbolo de aparecer en la secuencia de entrada. La suma de las amplitudes de los intervalos debe ser igual a 1. Previamente es necesario establecer un orden entre los símbolos. No es necesario seguir algún criterio especial para establecer un orden entre los símbolos del alfabeto fuente, pero el orden establecido debe ser conocido por el decodificador para poder hacer una correcta decodificación en la señal. En este caso, el orden queda definido por el alfabeto en sí mismo (de menor a mayor), no siendo necesario ningún tipo de reordenamiento de los valores.

Para codificar una secuencia se siguen los siguientes pasos:

1. Se selecciona el primer símbolo de la secuencia de entrada y se localiza el intervalo asociado a ese símbolo.

2. A continuación se selecciona el siguiente símbolo y se localiza su intervalo. Se multiplican los extremos de este intervalo por la longitud del intervalo asociado al símbolo anterior (es decir, por la probabilidad del símbolo anterior) y los resultados se suman al extremo inferior del intervalo asociado al símbolo anterior para obtener unos nuevos extremos inferior y superior. Para el símbolo i-ésimo se calcula su intervalo de la siguiente forma:

$$\begin{aligned} inf(i) &= inf(i-1) + (sup(i-1) - inf(i-1)) \times inf(i) \\ sup(i) &= inf(i-1) + (sup(i-1) - inf(i-1)) \times sup(i) \end{aligned}$$

3. El paso anterior se repite hasta que todos los símbolos del mensaje hayan sido procesados.

4. Por último se selecciona un valor dentro del intervalo del último símbolo de la secuencia. Este valor representará la secuencia que queremos enviar.

En principio, para la construcción de estos códigos es necesario tener un conocimiento previo de la frecuencia de ocurrencia de cada uno de los símbolos dentro de la secuencia de entrada, sin embargo, en este trabajo se utiliza una variante llamada *codificación aritmética adaptativa* en la cual al iniciar la codificación se presupone que todos los símbolos tienen la misma probabilidad de ocurrencia para luego modificar estas probabilidades con cada símbolo que se codifica.

El principio utilizado es muy simple: antes de comenzar a codificar la secuencia se inicializa un contador c_s para cada símbolo s del alfabeto Σ, estableciendo su valor en 1. La probabilidad de ocurrencia de cada símbolo se calcula en base a la cantidad de observaciones (capturadas por los contadores) según la fórmula 30. Luego de codificar cada símbolo, se incrementa el contador correspondiente en uno.

$$P_s = \frac{c_s}{\sum_{t \in \Sigma} c_t} \tag{30}$$

Durante la decodificación ocurre exactamente lo mismo: antes de comenzar se fija en 1 un contador de apariciones para cada símbolo del alfabeto, se decodifica el símbolo utilizando las ocurrencias para calcular los intervalos y, una vez que se conoce el símbolo decodificado, se incrementa el contador correspondiente.

8.4. Resultados de la codificación

La única etapa del proceso de codificación que implica pérdida de información es la cuantización de los bloques. La Tabla 18 muestra el *MSE* para cada fragmento de video luego de ser cuantizado.

Fragmento	*MSE*
P1_EXT_K1 - Fragmento 1	36,7658
P1_EXT_K4 - Fragmento 1	15,3548
P1_EXT_K4 - Fragmento 2	15,2352
P2_EXT2_K2-3 - Fragmento 1	19,6934
P2_EXT2_K2-3 - Fragmento 2	28,7357
Promedio	23, 1569

Tabla 18: MSE de cada fragmento luego de cuantizar los bloques de *Clase 2* y *Clase 3*

En relación a la codificación intermedia, la Tabla 19 muestra la cantidad de símbolos promedio utilizados para codificar cada cuadro.

La secuencia de bits obtenida del codificador aritmético es el resultado final producido por el método de compresión. Esto significa: partir una secuencia de video, aplicar el filtro de Wiener, luego el *algoritmo de selección de bloques para enviar* y la segunda clasificación de bloques, transformar con la DCT, cuantizar los bloques de *Clase 2* y *Clase 3*, luego representar el *bitmap* de clases, las matrices y vectores de desplazamiento correspondientes a cada cuadro mediante la codificación intermedia y finalmente comprimir esta última con un codificador aritmético.

Fragmento	Simbolos por Cuadro
P1_EXT_K1 - Fragmento 1	11.824
P1_EXT_K4 - Fragmento 1	4.573
P1_EXT_K4 - Fragmento 2	2.950
P2_EXT2_K2-3 - Fragmento 1	2.127
P2_EXT2_K2-3 - Fragmento 2	5.077
Promedio	5.310

Tabla 19: Promedio de símbolos por cuadro en la *Codificación Intermedia* de cada fragmento. La secuencia de pasos que seguidos hasta este punto son: filtro de Wiener, selección de bloques para enviar, segunda selección de bloques, DCT, cuantización y finalmente, codificación intermedia.

Las cadenas de bits resultantes son guardas en una archivos con la extensión *.SCVC* por las siglas en inglés de *compresor de video con cámara fija* o *Still Camera Video Compressor*. Analizando estas secuencias es posible determinar la eficiencia del método de compresión.

La Tabla 20 resume este análisis: la primera columna muestra el tamaño del archivo de salida por cada secuencia; la segunda, el *birate* o tasa de bits por segundo (calculado en base 29,97 cuadros por segundo); la tercera, la tasa de compresión obtenida (calculada a partir del tamaño de los fragmentos sin comprimir); y la última columna muestra la cantidad promedio de bits necesarios para codificar cada pixel del video.

Fragmento	Tamaño	Tasa de Comp.	Bitrate	Bits por Pixel
P1_EXT_K1 - Fragmento 1	2.671*KB*	44,3 : 1	444*Kbps*	0,1797
P1_EXT_K4 - Fragmento 1	1.219*KB*	97,0 : 1	203*Kbps*	0,0820
P1_EXT_K4 - Fragmento 2	501*KB*	236,1 : 1	83*Kbps*	0,0337
P2_EXT2_K2-3 - Fragmento 1	278*KB*	423,1 : 1	46*Kbps*	0,0187
P2_EXT2_K2-3 - Fragmento 2	1.160*KB*	101,9 : 1	193*Kbps*	0,0780
Promedio	1.165*KB*	180,5 : 1	194*Kbps*	0,0784

Tabla 20: Análisis de las secuencias de bits resultantes de la aplicación del método de compresión completo. Esto es, filtro de Wiener, selección de bloques para enviar, segunda clasificación de bloques, DCT, cuantización, codificación intermedia y compresión aritmética. La secuencia codificada incluye el *bitmap* de bloques, los vectores de movimiento y las matrices de cada bloque.

9. Resultados

En esta sección se analizan los resultados obtenidos. Se estudian los resultados de comprimir los 5 fragmentos de video utilizados a lo largo de todo el trabajo, así como también, los resultados de procesar las 6 secuencias de video completas. Para los fragmentos se comparan los resultados del método de compresión propuesto con los resultados obtenidos mediante los estándares MPEG-1 [2] y H.264/MPEG-4 AVC [24].

9.1. Comparación con MPEG-1 y H.264

Para poner en contexto la eficiencia del método de compresión presentado en este trabajo, sus resultados se comparan con dos de los métodos de compresión más utilizados:

Estándar MPEG-1, el cual, pese a haber sido desarrollado hace más de 20 años, continúa siendo uno de los formatos de video más utilizado en el mundo. Los videos utilizados en este trabajo fueron comprimidos originalmente utilizando este algoritmo.

Por otro lado, se analizan también los resultados obtenidos con el estándar H.264, el cual fue desarrollado en entre los años 2001 y 2003 con el objetivo de crear un estándar capaz de proporcionar una buena calidad de imagen con tasas binarias extremadamente bajas. Hoy en día el uso de este formato está en aumento pues forma parte de los formatos de video soportado por la norma BLU-RAY y sus aplicaciones se extienden desde la distribución de videos por la WEB hasta la captura de video en alta definición en dispositivos móviles.

En la sección anterior se mostraron resultados obtenidos procesar los fragmentos de video con el método de compresión propuesto (la Tabla 21 resume estos valores). Para que la comparación entre los distintos métodos fuera consistente, se re-codificaron los fragmentos de video a fin de obtener versiones de los mismos en formatos MPEG-2 y H.264 con tamaños y *bitrates* similares a los expuestos en la Tabla 21. Para ello se utilizó nuevamente la herramienta *Total Video Converter* y se ajusto manualmente el *bitrate* para cada fragmento hasta conseguir que los archivos de salida tuviesen el tamaño que cumpliera con el *bitrate* deseado.

Ya con los fragmentos re-codificados en ambos formatos, se utilizó un modelo de Simulink para compararlos cuadro a cuadro con los videos originales calculando su *MSE*. Los resultados obtenidos pueden verse en la Tabla 22. Esta tabla muestra para los tres formatos el tamaño del archivo de salida, el *bitrate* con el cual fue codificado el video y el *MSE* calculado como el promedio del *MSE* de cada cuadro. Notar que, a fin de facilitar la lectura, en esta tabla se llama al método de compresión presentado en este trabajo con la sigla *SCVC* (por la extensión elegida para los archivos de salida).

Fragmento	Tamaño	Tasa de Comp.	Bitrate	MSE
P1_EXT_K1 - Fragmento 1	2.671*KB*	44,3 : 1	444*Kbps*	36,7680
P1_EXT_K4 - Fragmento 1	1.219*KB*	97,0 : 1	203*Kbps*	15,3548
P1_EXT_K4 - Fragmento 2	501*KB*	236,1 : 1	83*Kbps*	15,2352
P2_EXT2_K2-3 - Fragmento 1	278*KB*	423,1 : 1	46*Kbps*	19,6934
P2_EXT2_K2-3 - Fragmento 2	1.160*KB*	101,9 : 1	193*Kbps*	28,7357
Promedio	1.165*KB*	180,5 : 1	194*Kbps*	23,1574

Tabla 21: Resumen de los resultados obtenidos al aplicar el método de compresión completo (filtro de Wiener, *algoritmo de selección de bloques para enviar*, segunda selección de bloques, DCT, cuantización, codificación intermedia y codificación aritmética) a los 5 fragmentos de video.

Fragmento	Formato	Tamaño	Bitrate	MSE
P1_EXT_K1 - Fragmento 1	*SCVC*	2.671*KB*	444*Kbps*	36,768
	MPEG-1	2.672*KB*	423*Kbps*	18,194
	H.264	2.683*KB*	419*Kbps*	13,053
P1_EXT_K4 - Fragmento 1	*SCVC*	1.219*KB*	203*Kbps*	15,354
	MPEG-1	1.222*KB*	161*Kbps*	12,752
	H.264	1.220*KB*	164*Kbps*	10,851
P1_EXT_K4 - Fragmento 2	*SCVC*	501*KB*	83*Kbps*	15,235
	MPEG-1	510*KB*	30*Kbps*	24,386
	H.264	508*KB*	35*Kbps*	22,393
P2_EXT2_K2-3 - Fragmento 1	*SCVC*	278*KB*	46*Kbps*	19,693
	MPEG-1	670*KB*	46*Kbps*	28,810
	H.264	245*KB*	46*Kbps*	29,016
P2_EXT2_K2-3 - Fragmento 2	*SCVC*	1.160*KB*	193*Kbps*	28,736
	MPEG-1	1.162*KB*	147*Kbps*	18,687
	H.264	1.162*KB*	145*Kbps*	16,066
Promedio	*SCVC*	1.166*KB*	193*Kbps*	23,157
	MPEG-1	1.247*KB*	161*Kbps*	20,566
	H.264	1.164*KB*	162*Kbps*	18,276

Tabla 22: Comparación de resultados entre el algoritmo de compresión presentado (indicado como *SCVC*) y los estándares MPEG-1 y H.264/MPEG-4 AVC.

En la Tabla 22 se observa que, en base al *MSE*, el rendimiento del algoritmo varía de una secuencia a otra. Más en detalle, para los fragmentos *P1_EXT_K1 - Fragmento 1*, *P1_EXT_K4 - Fragmento 1* y *P2_EXT2_K2-3 - Fragmento 2* el método aquí presentado parece tener un rendimiento marcadamente menor a MPEG-1 y más aún que H.264; mientras que, para los fragmentos *P1_EXT_K4 - Fragmento 2* y *P2_EXT2_K2-3 - Fragmento 1*, arroja mejores resultados que ambos estándares. No obstante, debido a la naturaleza de este trabajo, el análisis del *MSE* sólo sirve como referencia. A continuación se analizan uno a uno los resultados obtenidos no solo numéricamente sino también en base la calidad visual de las secuencias de video. Para facilitar la lectura, las imágenes correspondientes al análisis se encuentran en el Apéndice B.

9.1.1. Análisis de las secuencias: *P1_EXT_K1 - Fragmento 1*

La Figura 23 muestra la evolución cuadro a cuadro del *MSE*, calculado sobre los cuadros completos, para la secuencia *P1_EXT_K1 - Fragmento 1* comprimida con los tres métodos evaluados.

Al analizar los cuadros completos se observa que el rendimiento del algoritmo propuesto es claramente inferior (en promedio el *MSE* duplica al de los otros dos formatos). En particular, entre los cuadros 880 y 1020 se observan dos picos en el gráfico. Éstos coinciden con el momento en que un médico muestra al paciente algo escrito en un block de hojas. Cuando el block entra en la escena la cámara ajusta automáticamente la apertura de la lente (para compensar el cambio de luminosidad), oscureciendo así el fondo de la escena. Como el cambio de luminosidad es gradual, el algoritmo tarda en adaptarse y genera un efecto de cuadriculado (tal como se muestra en la Figura 24), que es la causa de los picos en el *MSE*.

Al restringir el análisis sólo a las regiones de la escena que aportan información relevante para el *observador* (mediante la aplicación de una máscara sobre el video que sólo conserva el rostro del paciente), se advierte que, si bien la diferencia en calidad se mantiene, como se puede observar en la Figura 25 los picos entre los cuadros 880 y 1020 desaparecen.

Pese a las grandes diferencias en el *MSE*, al realizar un análisis visual de los videos se notó que las imágenes producidas por el método propuesto son nítidas y los rasgos del paciente se distinguen con claridad. En la Figura 26 se puede observar el cuadro 400 del video original y su equivalente para cada uno de los métodos evaluados. La segunda fila de imágenes corresponde a una ampliación de la boca del paciente, en donde se puede observar que el algoritmo de compresión propuesto tiene éxito al preservar los detalles finos, incluso genera imágenes con más nivel de detalle que MPEG-1.

9.1.2. Análisis de las secuencias: *P1_EXT_K4 - Fragmento 1*

En base a la Tabla 22, el método de compresión propuesto en este trabajo parece estar por debajo de MPEG-1 y H.264, sin embargo, al analizar la evolución del *MSE* a lo largo de toda la secuencia observamos que esto no es necesariamente cierto.

Como se observa en la Figura 27, inicialmente el *MSE* para el algoritmo propuesto es de alrededor de 15 y se mantiene casi constante hasta el cuadro 817. A su vez, hasta este punto, el *MSE* para los otros dos métodos va incrementando lentamente. A partir del cuadro 817 los valores comienzan a cambiar bruscamente. Para comprender las razones se debe analizar qué sucede a partir de la segunda mitad del video.

Hasta el cuadro 817 el paciente se encuentra durmiendo, los cambios en la escena son mínimos, sólo se observan pequeños movimientos producto de la respiración del paciente. En este punto el paciente sufre un ataque, se despierta y comienza a moverse. Estos movimientos provocan variaciones en la calidad de la imagen en los tres métodos.

Más adelante, en el cuadro 1141 el operador de la cámara ajusta el zoom de modo que la escena pasa rápidamente de enfocar el torso del paciente a enfocar su cuerpo completo, luego el rostro y luego, nuevamente, su cuerpo entero. El método se adapta a estos cambios bruscos en la escena generando un gran número de bloques de *Clase 3* que rápidamente reconstruyen la imagen. Sin embargo, debido al bajo *bitrate*, tango MPEG-1 como H.264 no pueden codificar la información necesaria para representar los cambios en la escena y, como consecuencia, la calidad de la imagen disminuye drásticamente.

Como se mencionó anteriormente, la información más valiosa para los especialistas está dada por los movimientos sutiles en el rostro del paciente durante los ataques. Si esta información se pierde debido a la compresión, el video se torna virtualmente inútil. En la Figura 28 (dónde se muestra el cuadro 1331 del video) es posible observar que MPEG-1 y H.264 distorsionan la imagen al punto que el rostro y las manos del paciente son casi indistinguibles.

Es posible afirmar que para este video, pese a que matemáticamente MPEG-1 y H.264 generan imágenes de mejor calidad, el algoritmo propuesto genera videos que resultan de mucha más utilidad para el diagnóstico que los anteriores.

9.1.3. Análisis de las secuencias: *P1_EXT_K4 - Fragmento 2*

Durante toda esta secuencia, el paciente se encuentra en reposo, de modo que los únicos movimientos que se perciben se deben a su respiración. Gracias a esto el el algoritmo desarrollado logra obtener una alta tasa de compresión (de 236:1) con poca pérdida de calidad.

Si se observa la Figura 29 se ve que mientras el *MSE* para el CODEC propuesto se mantiene constante durante todo el video, tanto MPEG-1 como H.264 presentan una constante perdida de calidad. Lo que sucede es que mientras el primero utiliza una matriz de cuantización constante durante todo el video, MPEG-1 y H.264 adaptan sus matrices en base al contenido de los cuadros. Como los cuadros son todos muy similares entre sí, MPEG-1 y H.264 gradualmente aumentan los intervalos de cuantización para descartar más información y así alcanzar los bajos *bitrates* exigidos ($30Kbps$ para MPEG-1 y $35Kbps$ para H.264), esto trae como consecuencia la constante pérdida de calidad que se aprecia en el gráfico. En la Figura 30 se pueden ver distintos cuadros a lo largo del video para los tres métodos de compresión.

La perdida de calidad durante los períodos en que el paciente está en reposo no representa, en sí misma, un inconveniente para el diagnóstico. El problema se da durante la etapa inicial de los ataques: los ataques de epilepsia suelen comenzar con movimientos sutiles en el rostro y manos del paciente luego de un período de reposo, si el algoritmo de compresión descarta demasiada información durante estos períodos, se corre el riesgo de perder los detalles del inicio de un ataque.

9.1.4. Análisis de las secuencias: *P2_EXT2_K2-3 - Fragmento 1*

Al momento de recodificar esta secuencia en formato MPEG-1, no fue posible obtener una secuencia que tuviese un tamaño similar a la obtenida con el *CODEC* aquí propuesto. Fue por esto que se decidió codificarla con un *bitrate* de $46Kbps$, esto genera un archivo de $670KB$, mucho más que los $278KB$ obtenidos el algoritmo propuesto (con un *bitrate* similar). Con H.264, sin embargo, fue posible obtener un archivo de salida de $245KB$.

La evolución del *MSE* cuadro a cuadro se puede ver en la Figura 31. Durante la primera mitad del video, el paciente se encuentra en reposo, casi sin moverse. Durante este intervalo, la calidad de la imagen se mantiene casi constante para el algoritmo propuesto y lo mismo sucede para H.264, aunque su *MSE* es claramente más bajo. Para MPEG-1, sin embargo, el *MSE* aumenta gradualmente hasta el punto en que, en el cuadro 718, supera al de SCVC.

Entre los cuadros 940 y 1064 el paciente lleva su mano hacia su boca para luego bajarla. Nuevamente, debido al bajo *bitrate*, estos cambios en la escena no pueden ser representados correctamente por MPEG-1 y H.264, lo que trae como consecuencia una marcada caída en la calidad de la imagen. Luego del cuadro 1064 no hay movimientos marcados, pero pese a esto MPEG-1 y H.264 no logran mejorar la calidad de la imagen.

La Figura 32 muestra cuatro cuadros del video en los que se puede apreciar cómo varía la calidad de la imagen a lo largo del tiempo.

9.1.5. Análisis de las secuencias: *P2_EXT2_K2-3 - Fragmento 2*

Para este fragmento, la comparación del *MSE* arroja que, en promedio, tanto MPEG-1 como H.264 superan en calidad al método propuesto. Al analizar el gráfico de la Figura 33 se puede ver que la calidad de las secuencias resultantes de los tres métodos varía marcadamente a lo largo del tiempo y, en algunos puntos, el método presentado en este trabajo supera en calidad a los otros dos.

Esta secuencia de video corresponde a una serie de preguntas y exámenes físicos que un especialista le realiza al paciente luego de que este sufriese un ataque, por lo tanto cada detalle de la escena es de gran importancia para el diagnóstico médico.

La secuencia intercala períodos de reposo con movimientos del paciente, lo que sigue es una pequeña crónica de lo que puede verse en el video:

- Entre los cuadros 0 y 50 el paciente mueve su brazo izquierdo.
- Entre los cuadros 50 y 196 el paciente habla pero sin moverse.
- Entre los cuadros 196 y 286, una enfermera acomoda los cables del electroencefalógrafo.
- Entre los cuadros 286 y 390, el paciente sigue respondiendo preguntas sin moverse.
- Entre los cuadros 390 y 480, se cubre la boca con su mano derecha y luego la baja.
- Entre los cuadros 538 y 595, el médico le muestra al paciente una lapicera y le pide que la tome.
- Entre los cuadro 595 y 695, el paciente toma la lapicera con su mano derecha.
- Entre los cuadros 695 y 1015, el paciente se mantiene hablando y realiza algunos movimientos con sus mantos.
- Entre los cuadros 1015 y 1056, devuelve la lapicera.
- Finalmente, entre los cuadros 1056 y 1145, el paciente levanta ambos brazos y luego los baja.

Al analizar lo que sucede con el *MSE* en los períodos de movimiento y de reposo, se observó que durante los períodos de reposo el *MSE* de SCVC se incrementa y cuando hay movimiento, disminuye. Exactamente lo contrario ocurre con MPEG-1 y H.264. En particular los cuadros 260 y 1063, que corresponden con los momentos en que los movimientos son más rápidos y más cambios se producen

en la escena, coinciden con los picos de mayor *MSE* para MPEG-1 y H.264 y de menor *MSE* para el método propuesto.

Esto se debe a que, durante los períodos de reposo, el *algoritmo de selección de bloques para enviar* clasifica una gran cantidad de bloques como *no relevantes*, estos bloques son luego clasificados como *Clase 0* y no son codificados. Como consecuencia la calidad de la imagen disminuye y esto aumenta el *MSE*. Cuando la escena cambia rápidamente el *algoritmo de selección de bloques para enviar* clasifica más bloques como *relevantes* y como consecuencia la calidad de la imagen aumenta.

Por el contrario, debido al bajo *bitrate*, MPEG-1 y H.264 no tienen problema para codificar los períodos con poco movimiento, pero cuando ocurren cambios rápidos, no son capases de codificar la cantidad de información necesaria para mantener la calidad de la imagen. La Figura 34 muestra cuatro cuadros del video, dos en períodos de reposo (330 y 1332) y dos con movimiento (265 y 1061).

9.2. Secuencias completas

Como se mencionó en la Sección 4 hasta este punto se ha trabajado con fragmentos cortos de video (1438 cuadros), extraídos de los 6 videos provenientes de estudios de *Video-EGG* que se listan en la Tabla 1. A continuación se analizan los resultados obtenidos al aplicar el método de compresión presentado en este trabajo a estos videos completos (que se muestran en la Tabla 23) y se realiza una comparación visual respecto de los videos obtenidos con los estándares MPEG-1 y H.264.

Video	Tamaño	Tasa de Comp.	Bitrate	MSE
P1_EXT_K1	$2.935KB$	45,2 : 1	$438Kbps$	36,613
P1_EXT_K3	$16.091KB$	38,5 : 1	$513Kbps$	47,502
P1_EXT_K4	$12.553KB$	92,1 : 1	$214Kbps$	17,699
P2_EXT2_K2-3	$12.222KB$	163,4 : 1	$121Kbps$	28,394
P3_EXT8_K2-3	$39.576KB$	50,1 : 1	$395Kbps$	35,848
P3_EXT10_K4	$36.641KB$	35,2 : 1	$561Kbps$	24,708
Promedio	$20.003KB$	70,8 : 1	$374Kbps$	31,794

Tabla 23: Resumen de los resultados obtenidos al comprimir las secuencias de video completas presentadas en la Sección 4.1.

Siguiendo el mismo procedimiento que en la sección anterior, se re-codificaron los videos originales con los *CODEC*s MPEG-1 y H.264, de modo tal que todos videos resultantes para cada secuencia tuviesen tamaños similares.

9.2.1. Análisis de las secuencias: P1_EXT_K1

El video *P1_EXT_K1* es, básicamente, la secuencia *P1_EXT_K1 - Fragmento 1* pero con 5 segundos más de duración. Por esta razón los resultados para ambas secuencias son prácticamente idénticos. La Figura 35 muestra la evolución del *MSE* para este video.

Al igual que en *P1_EXT_K1 - Fragmento 1*, los picos en los cuadros 910 y 1006 que se observan en el gráfico coinciden con el momento en que alguien interpone un block de hojas entre la cámara y el paciente. El efecto que el cambio de luminosidad tiene en la imagen se puede observar en la Figura 36.

Independientemente del *MSE*, la calidad visual de este video es muy buena. La única distorsión importante se observa entre los cuadros mencionados anteriormente, sin embargo, esto no representa ningún problema para realizar el diagnóstico médico.

En la Figura 36 comparamos varios cuadros del video procesado con el video original y los obtenidos mediante MPEG-1 y H.264.

9.2.2. Análisis de las secuencias: P1_EXT_K3

En este video sólo se observa el torso del paciente mientras está acostado, al principio leyendo una revista y luego sufre un ataque. Durante el ataque se le hacen una serie de preguntas, se le muestran distintos objetos y se le piden que mueva los brazos en varias oportunidades. Este video contiene al video *P1_EXT_K1* y si se observa la evolución del *MSE* que se muestra en la Figura 45 es posible notar el mismo pico que se observa para dicho video.

La calidad visual es muy buena, aunque esporádicamente se puede observar un efecto de pixelado. Entre los cuadros 3000 y 3265, el cambio de luminosidad de la escena hace que la distorsión de pixelado cubra gran parte del fondo de la escena (lo cual provoca el pico de *MSE* que se observa en el gráfico), pero no afecta a la imagen del paciente.

Al comparar los resultados obtenidos con la misma secuencia comprimida utilizando MPEG-1 y H.264 observamos que la calidad de la imagen es similar. La Figura 46 muestra la comparación entre los distintos métodos de compresión para 4 cuadros de la secuencia. En algunos puntos del video, en donde el paciente realiza algún movimiento rápido (ver cuadros 72 y 6089) se observa que MPEG-1 y H.264 degradan ligeramente la calidad de la imagen, pero esta perdida de calidad sólo se extiendo durante unos pocos cuadros.

9.2.3. Análisis de las secuencias: P1_EXT_K4

Tal como sucedía con varios de los fragmentos estudiados anteriormente, la calidad del video resultante de aplicar el método de compresión a la secuencia

P1_EXT_K4 varía marcadamente dependiendo de lo que sucede en la escena. El gráfico de la Figura 37 se observa que el *MSE* oscila continuamente tomando valores entre 12,5 y 23,8. Estas variaciones se deben a la velocidad con la que se producen los cambios en el video: cuando los cambios son rápidos la calidad aumenta y cuando éstos son lentos (o casi no hay cambios en la escena) la calidad baja.

La Figura 38 muestra 4 cuadros del video: dos con *MSE* cercano a la media (cuadros 1576 y 11891); uno correspondiente al momento de mínimo *MSE* (cuadro 4157); y uno con *MSE* máximo (cuadro 7172).

Al comparar los resultados con los videos codificados con MPEG-1 y H.264, se notó que cuando la cámara se mueve o el paciente realiza movimientos rápidos, la imagen producida por estos se *pixela* notablemente y no es posible ver en detalle el rostro del paciente. Esto podemos verlo, por ejemplo, en el cuadro 11891 de la Figura 38.

9.2.4. Análisis de las secuencias: P2_EXT2_K2-3

Para este video también se observan marcadas variaciones en el *MSE*. Nuevamente, los intervalos con menor calidad corresponden a los períodos de reposo y los de menor *MSE* a los momentos con más movimiento. La Figura 39 muestra la evolución del *MSE* en función del tiempo.

Con este video en particular sucede que durante los períodos de reposo la imagen se degrada notablemente, llegando al punto en que no se distinguen los rasgos de rostro del paciente. Esto se debe a que el paciente se mueve muy lentamente y esto causa que el *algoritmo de selección de bloques para enviar* interprete su rostro como parte del *fondo* de la escena. Sin embargo, entre los cuadros 6000 y 7500 el paciente sufre un ataque y, debido a los movimientos rápidos que éste realiza, la calidad del video se incrementa permitiendo apreciar en detalle toda la escena.

Al analizar los videos resultantes de MPEG-1 y H.264, encontramos que debido al bajo *bitrate* (121Kbps), los mismos tienen buena calidad visual durante los períodos de reposo, pero entre los cuadros 6000 y 7500 la calidad de la imagen se degrada notablemente. La Figura 40 muestra cuatro cuadros del video *P2_EXT2_K2-3* procesados con los distintos algoritmos.

9.2.5. Análisis de las secuencias: P3_EXT8_K2-3

Este video también intercala períodos de reposo con segmentos con mucho movimiento (el paciente sufre dos ataques), lo cual se refleja en las oscilaciones del *MSE* que se aprecian en la Figura 43.

Visualmente la calidad es comparable a MPEG-1 y H.264 excepto en los momentos de mayor movimiento, cuando la calidad de estos últimos métodos decrece debido al bajo *bitrate*. La Figura 44 muestra dos cuadros con alto *MSE* (543 y 19310) y otros dos con valores bajos (9993 y 18766).

9.2.6. Análisis de las secuencias: P3_EXT10_K4

La tasa de compresión conseguida para este video fue la menor de todas. Esto se debe, principalmente, a la mala iluminación, pues la luz parpadea constantemente, produciendo constantes cambios en la escena y forzando al algoritmo a codificar una gran cantidad de información innecesaria. Como contraparte, el *MSE* para esta secuencia es el menor obtenido. La figura 41 muestra la evolución del *MSE* en función del tiempo.

La calidad visual de esta secuencia es muy buena, comparable a los resultados obtenidos con MPEG-1 y H.264. El único momento del video en que se pueden apreciar distorsiones es entre los cuadros 10300 y 11000, en dónde se genera un efecto de cuadriculado sobre la pierna del paciente. En Figura 42 muestra cuatro cuadros del video: el cuadro 3531, con *MSE* promedio; los cuadros 8562 y 10220, correspondientes a los dos valores de *MSE* mínimos; y el cuadro 10690 correspondiente al momento con mayor *MSE*.

9.3. Relación entre el MSE y el Filtro de Wiener

Como se mencionó en la sección 6, para mejorar el rendimiento del *algoritmo de selección de bloques para enviar*, se aplica a las secuencias de video el filtro de Wiener. Esto reduce el nivel del *ruido*, pero introduce cierto nivel de error que, más tarde, se ve reflejado en el *MSE* de los resultados. Es difícil decir cuánto del *MSE* de una secuencia comprimida se debe al filtro de Wiener, pero sin duda éste determina la cota máxima para la calidad de los resultados.

La Figura 22 muestra el *MSE* de cada uno de los fragmentos de video luego de ser procesados con el filtro de Wiener. La Tabla 24 permite comparar el *MSE* promedio de cada secuencia comprimida con el error introducido por el filtro de Wiener.

Fragmento	MSE Comprimidas	MSE Wiener
P1_EXT_K1 - Fragmento 1	36,7680	19,1161
P1_EXT_K4 - Fragmento 1	15,3548	4,7970
P1_EXT_K4 - Fragmento 2	15,2352	5,2511
P2_EXT2_K2-3 - Fragmento 1	19,6934	7,6312
P2_EXT2_K2-3 - Fragmento 2	28,7357	7,6653

Tabla 24: Comparación del *MSE* promedio para cada fragmento comprimido con el error generado al aplicar el filtro de Wiener.

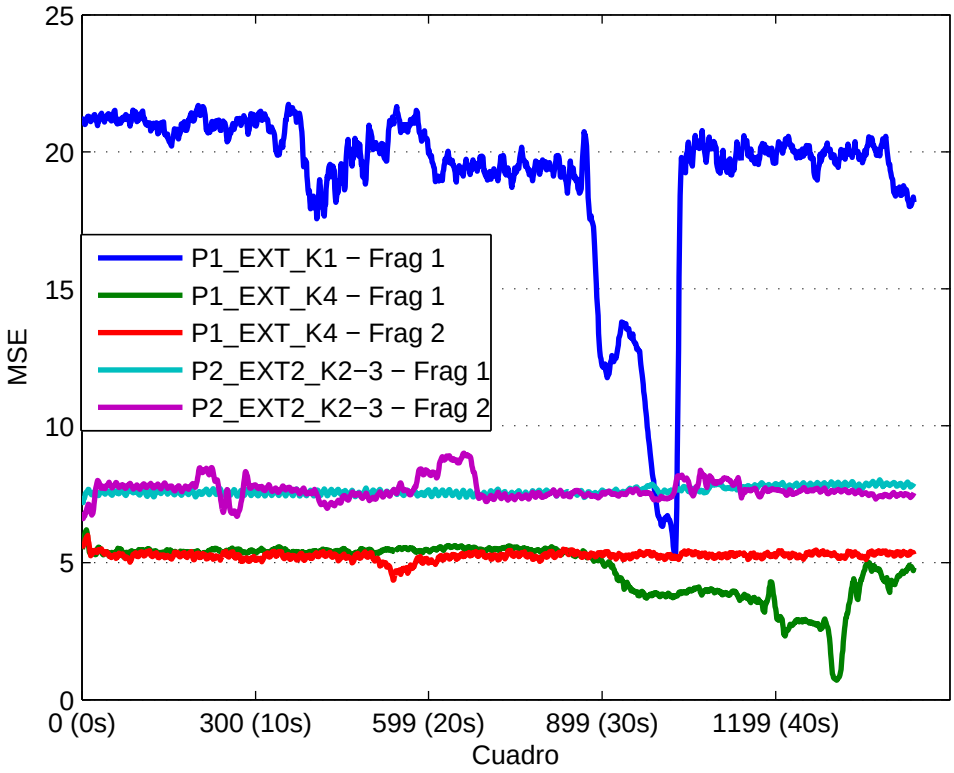


Figura 22: *MSE* introducido por el filtro de Wiener para los 5 fragmentos de video

10. Conclusiones

El método de compresión presentado en este trabajo demostró cumplir satisfactoriamente con los objetivos propuestos. Los videos comprimidos, en general, logran preservar los detalles del rostro y el cuerpo de los pacientes, en especial durante el transcurso de las crisis. En contraste, durante los períodos en los que el paciente está en reposo o durmiendo, suelen aparecer un gran número de distorsiones; sin embargo esto no representa ningún problema ya que, por lo general, estos períodos de reposo no aportan información relevante para el diagnóstico médico. Esto se debe en gran parte al *algoritmo de selección de bloques para enviar*, el cual logra una buena separación entre el *plano principal* y el *fondo* de la escena, aún cuando los videos originales presentan un alto nivel de ruido.

Al analizar los resultados obtenidos, se observó que, cuando la tasa de compresión obtenida es inferior a 100 : 1 la calidad de los videos resultantes es comparable con MPEG-1 y H.264. A su vez, cuando se consiguen tasas superiores, se logra superar ampliamente en calidad a ambos estándares.

Las tasas de compresión más altas (superiores a 400 : 1) se obtuvo al comprimir segmentos de video con poco movimiento, sin que esto se traduzca en una marcada pérdida de calidad. También se observó que en estas condiciones los videos comprimidos con MPEG-1 y H.264 pierden calidad rápidamente al pasar de períodos de reposo a períodos de movimiento.

El ajuste de parámetros y umbrales puede mejorarse en investigaciones futuras: existen al menos 10 parámetros de cuyos valores dependen la tasa de compresión que se puede obtener y la calidad de los videos resultantes. En este trabajo se realizaron todas las mediciones utilizando el mismo conjunto de umbrales, pero se podrían conseguir mejores resultados si se ajustasen los valores para cada video. El ajuste automático de umbrales en base a las características del video es un ítem pendiente y podría ser objeto de futuras investigaciones.

A. Algoritmo de selección de bloques para enviar

Input: S la secuencia de video en escala de grises procesada con el filtro de Wiener.

Constatns: h_μ un umbral para la media, h_σ un umbral para el desvío estándar y R la cantidad de bloques por cuadro.

Output: Una secuencia T donde cada cuadro es un par (D, Q). D es la imagen del cuadro resultante del algoritmo y Q es una matriz con un componente por bloque indicando con *VERDADERO* si un bloque es *relevante* y con *FALSO* en caso contrario.

```
1  begin
2      Q_b^[0] ← VERDADERO for 0 ≤ b < R ;
3      D^[0] ← S^[0];
4      T^[0] ← (D^[0], Q^[0]) ;
5      K_b ← 0 for 0 ≤ b < R ;
6      for 1 ≤ k < #(S) do
7          for 0 ≤ b < R do
8              if σ(S_b^[k−1] − S_b^[k]) > h_σ ∨ μ(μ(S_b^[K_b:k]) − S_b^[c]) > h_μ then
9                  D_b^[k] ← S_b^[k];
10                 Q_b^[k] ← VERDADERO ;
11                 k_b ← k ;
12             else
13                 D_b^[k] ← S_b^[k_b] ;
14                 Q_b^[k] ← FALSO ;
15             end
16             T^[k] ← (D^[k], Q^[k]) ;
17         end
18     end
19 end
```

Algorithm 1: Algoritmo de selección de bloques para enviar.

A.1. Descripción del Algoritmo

El *algoritmo de selección de bloques para enviar* toma como entrada la secuencia S y produce como salida una secuencia T donde cada cuadro es un par (D, Q). D es la imagen del cuadro resultante del algoritmo y Q es una matriz booleana con una componente por bloque donde *VERDADERO* indica que un bloque es *relevante* y *FALSO* indica lo contrario (un ejemplo de esta matriz Q se muestra en la fila superior de la figura 9).

El proceso de inicialización del algoritmo es simple: En las líneas 1 a 4 se inicializan Q, D y T de modo tal que el primer cuadro de T sea exactamente igual al primer cuadro de S y que todos los bloques del mismo sean clasificados como *relevantes*. En la línea 5 se establece este cuadro como *cuadro clave* para todos los bloques. En todos los casos, R representa la cantidad de bloques en que se divide un cuadro.

Luego, los ciclos en las líneas 6 y 7 iteran sobre cada bloque b de cada cuadro c de S (omitiendo el primer cuadro que ya fue procesado durante la inicialización).

En la línea 8 se verifica las condiciones 16 y 17 sobre b. Si el bloque satisface alguna de ellas, se lo cataloga como *relevante* y se lo incluye como información nueva en el cuadro de salida (línea 9), se lo incluye con valor *VERDADERO* en la máscara de bloques Q (línea 10) y, a su vez, se define al cuadro c como *cuadro clave* para el bloque b.

Si las condiciones 16 y 17 no se cumplen para el bloque b, se lo reemplaza por el mismo bloque en el último *cuadro clave* y se coloca el valor *FALSO* en la máscara de bloques Q (líneas 13 y 14). Finalmente en la línea 16 se utilizan los resultados obtenidos para todos los bloques del cuadro para definir el cuadro c de la secuencia de salida T.

B. Imágenes y gráficos de los resultados

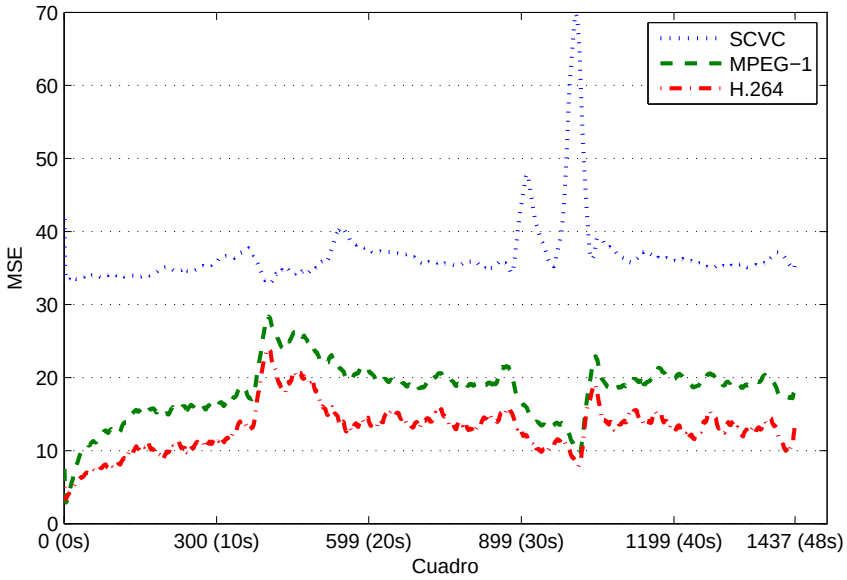
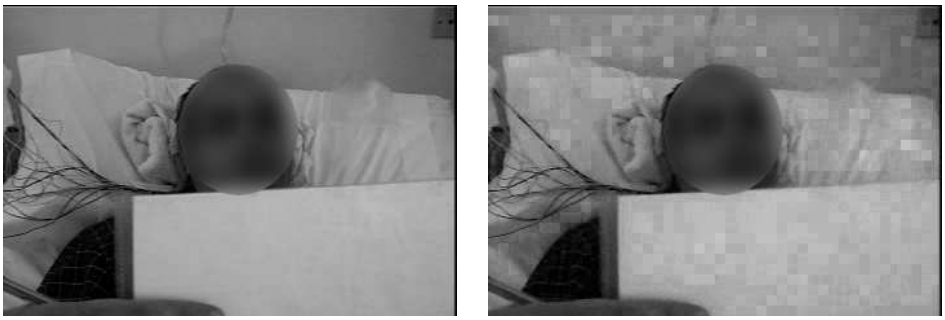


Figura 23: Evolución del *MSE* cuadro a cuadro para la secuencia *P1_EXT_K1 - Fragmento 1*



Original *SCVC*

Figura 24: Cuadro 1000 de la secuencia *P1_EXT_K1 - Fragmento 1*. Se observa el pixelado que aparece cuando la luminosidad de la escena disminuye gradualmente.

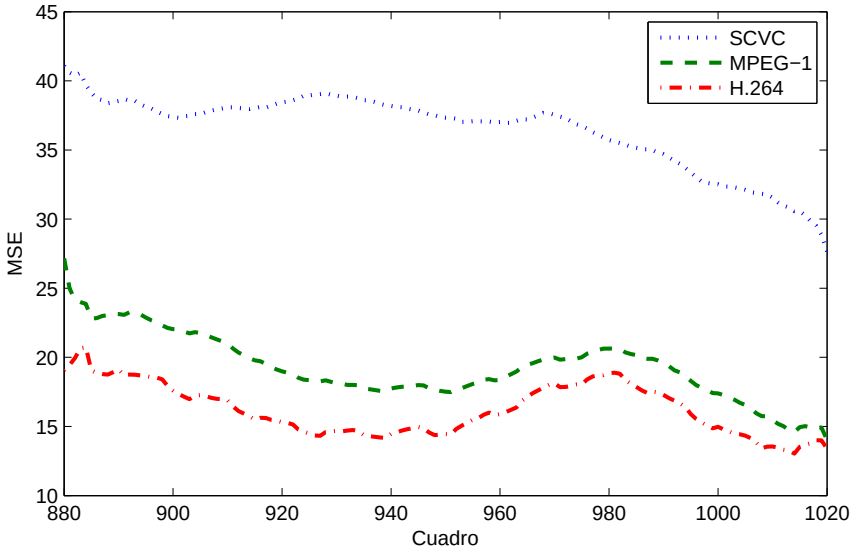


Figura 25: Evolución del *MSE* entre los cuadros 880 y 1020 de la secuencia *P1_EXT_K1 - Fragmento 1*, evaluado sólo sobre las zonas de la escena que resultan de interés para el *observador*.

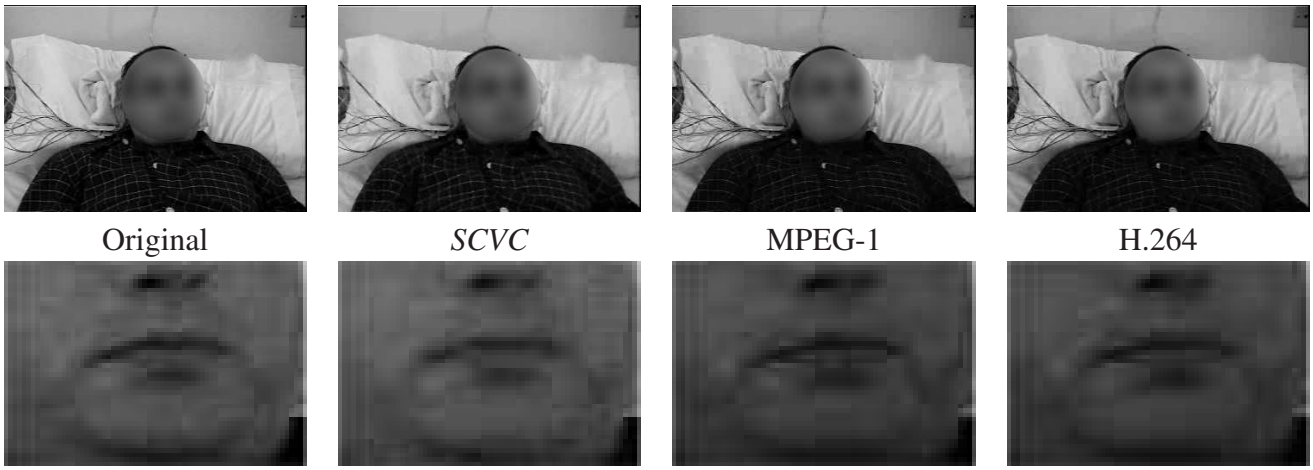


Figura 26: Arriba, cuadro 400 del video *P1_EXT_K1 - Fragmento 1* original y procesado por los tres algoritmos de compresión evaluados. Abajo, ampliación sobre la boca del paciente.

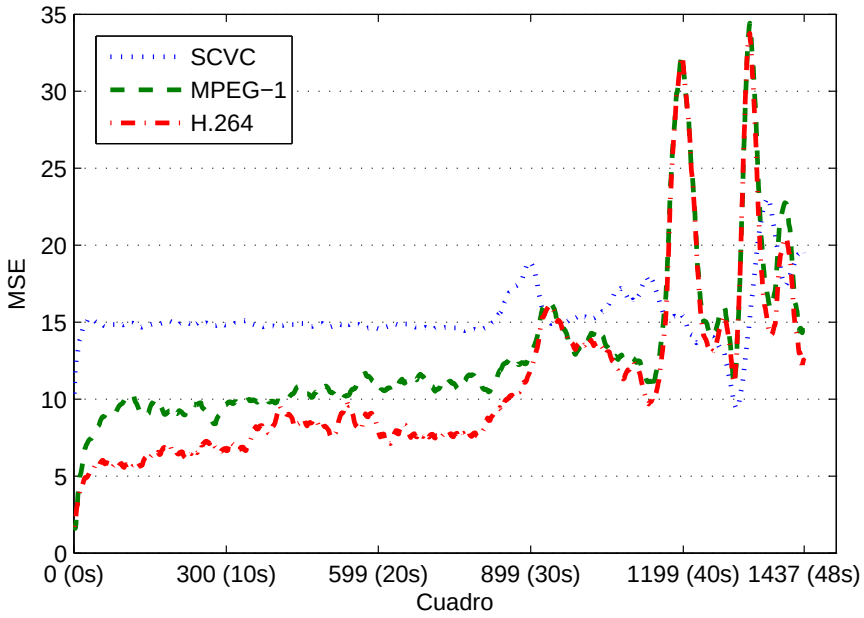


Figura 27: Evolución del *MSE* cuadro a cuadro para la secuencia *P1_EXT_K4 - Fragmento 1*

Original *SCVC* MPEG-1 H.264

Figura 28: Cuadro 1331 de la secuencia *P1_EXT_K4 - Fragmento 1* original y procesado por los tres métodos de compresión analizados.

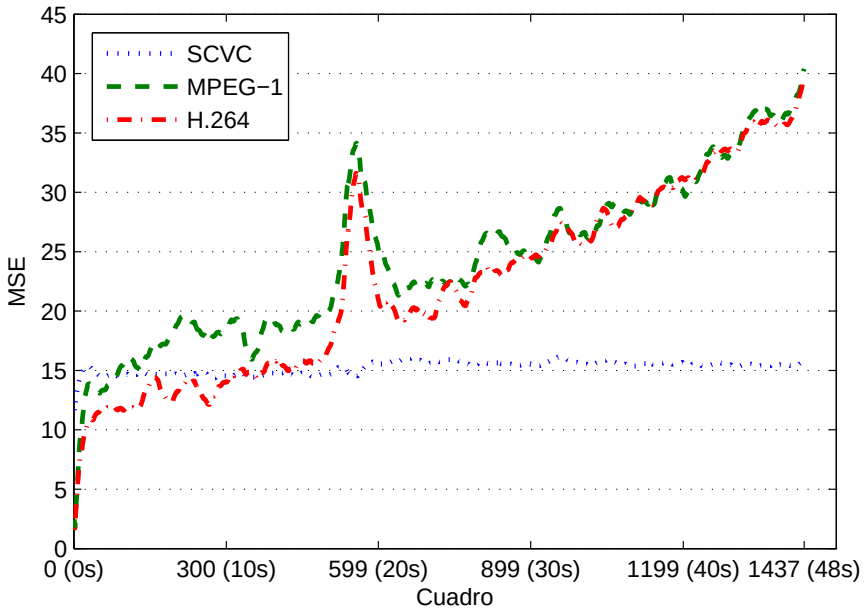


Figura 29: Evolución del *MSE* cuadro a cuadro para la secuencia *P1_EXT_K4 - Fragmento 2*

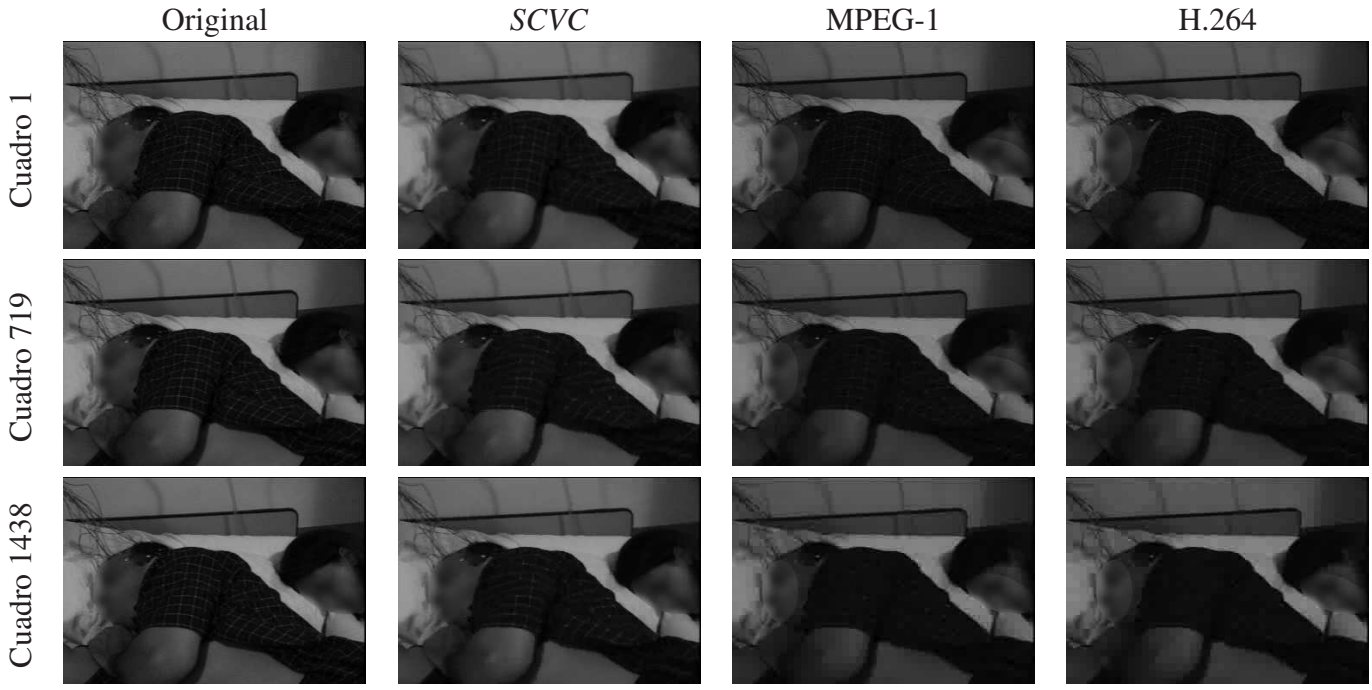


Figura 30: Cuadros 1, 719 y 1438 de la secuencia *P1_EXT_K4 - Fragmento 2* original y procesado por los tres métodos de compresión analizados.

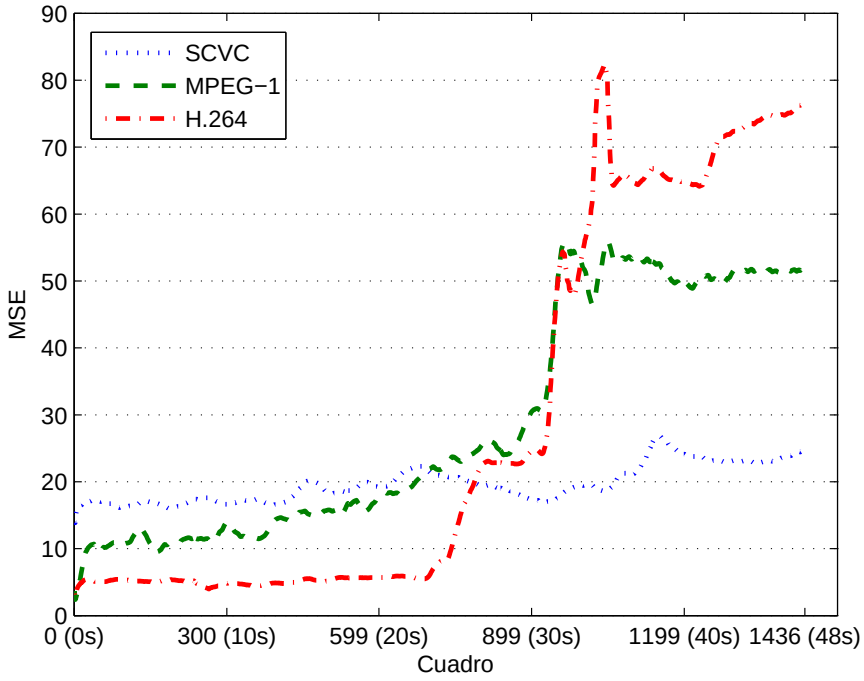


Figura 31: Evolución del *MSE* cuadro a cuadro para la secuencia *P2_EXT2_K2-3 - Fragmento 1*

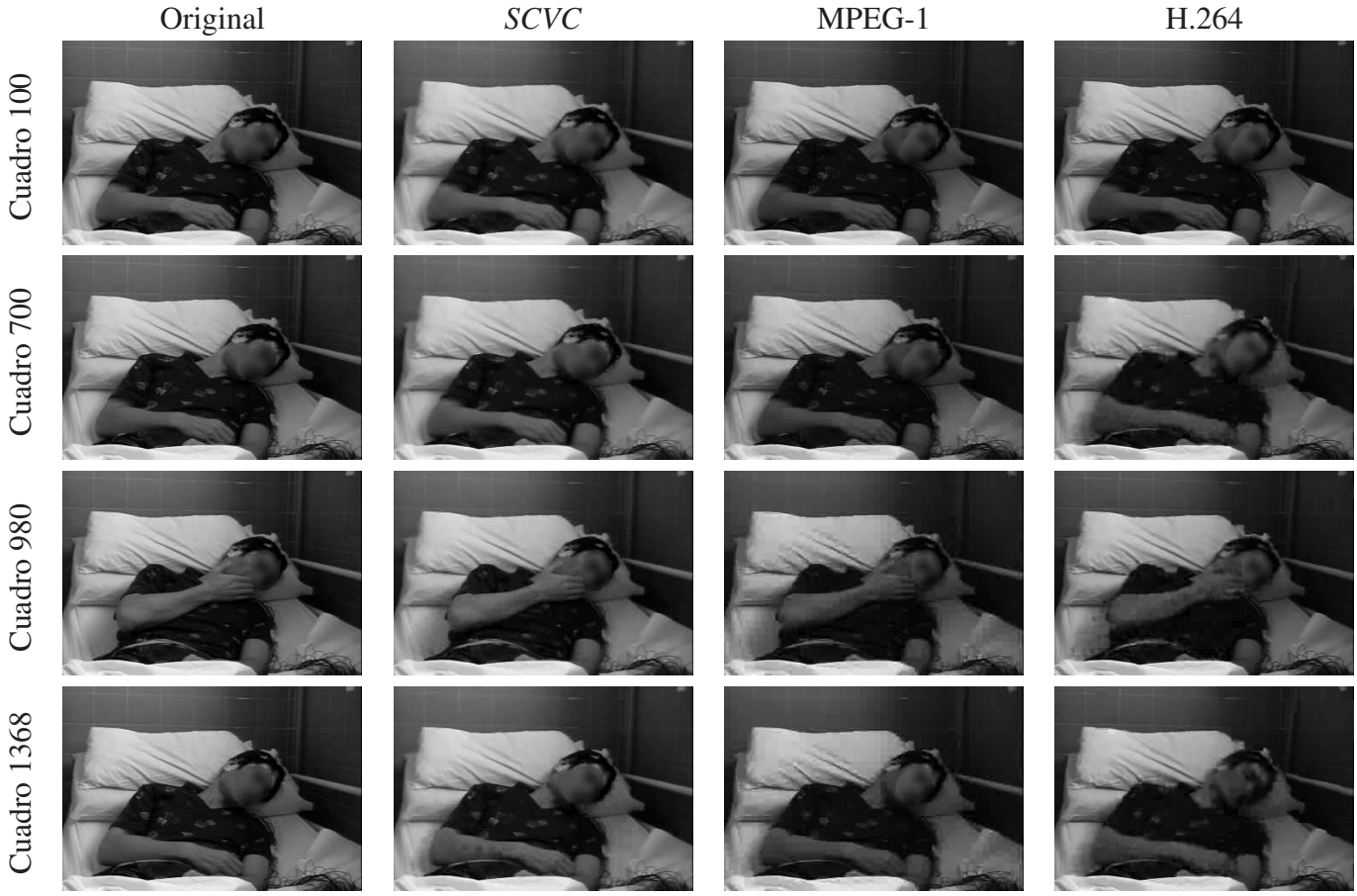


Figura 32: Cuadros 100, 700, 980 y 1368 de la secuencia *P2_EXT2_K2-3 - Fragmento 1* original y procesado por los tres métodos de compresión analizados.

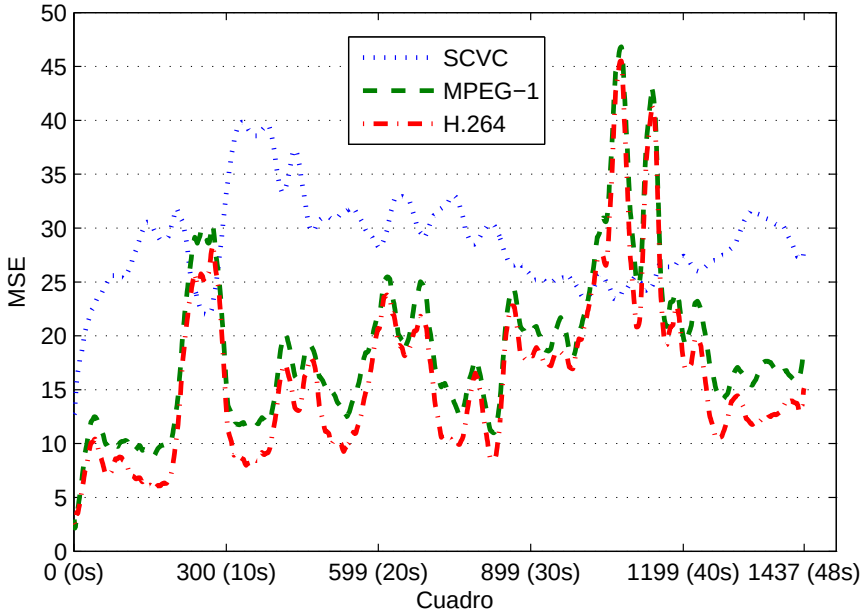


Figura 33: Evolución del *MSE* cuadro a cuadro para la secuencia *P2_EXT2_K2-3 - Fragmento 2*

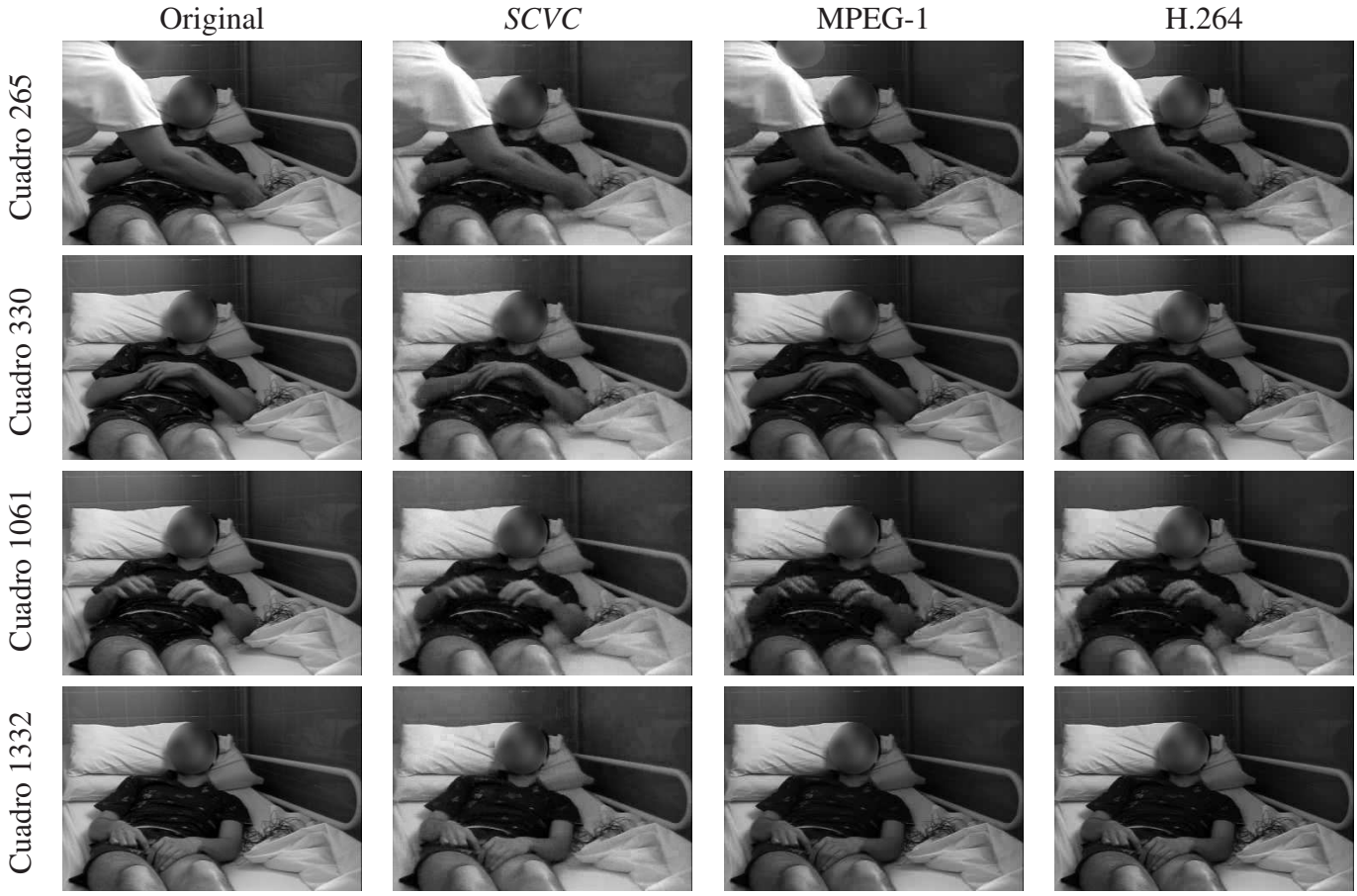


Figura 34: Cuadros 265, 330, 1061 y 1332 de la secuencia *P2_EXT2_K2-3 - Fragmento 2* original y procesado por los tres métodos de compresión analizados.

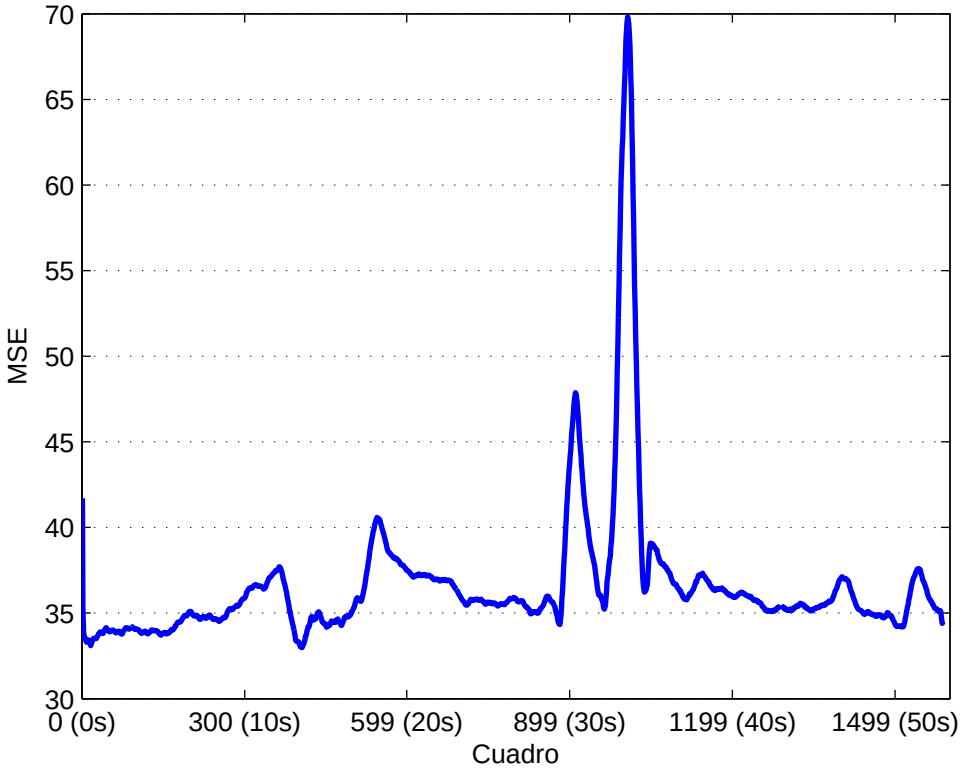


Figura 35: Evolución del *MSE* en función del tiempo para el video *P1_EXT_K1* comprimido con el método presentado en este trabajo.

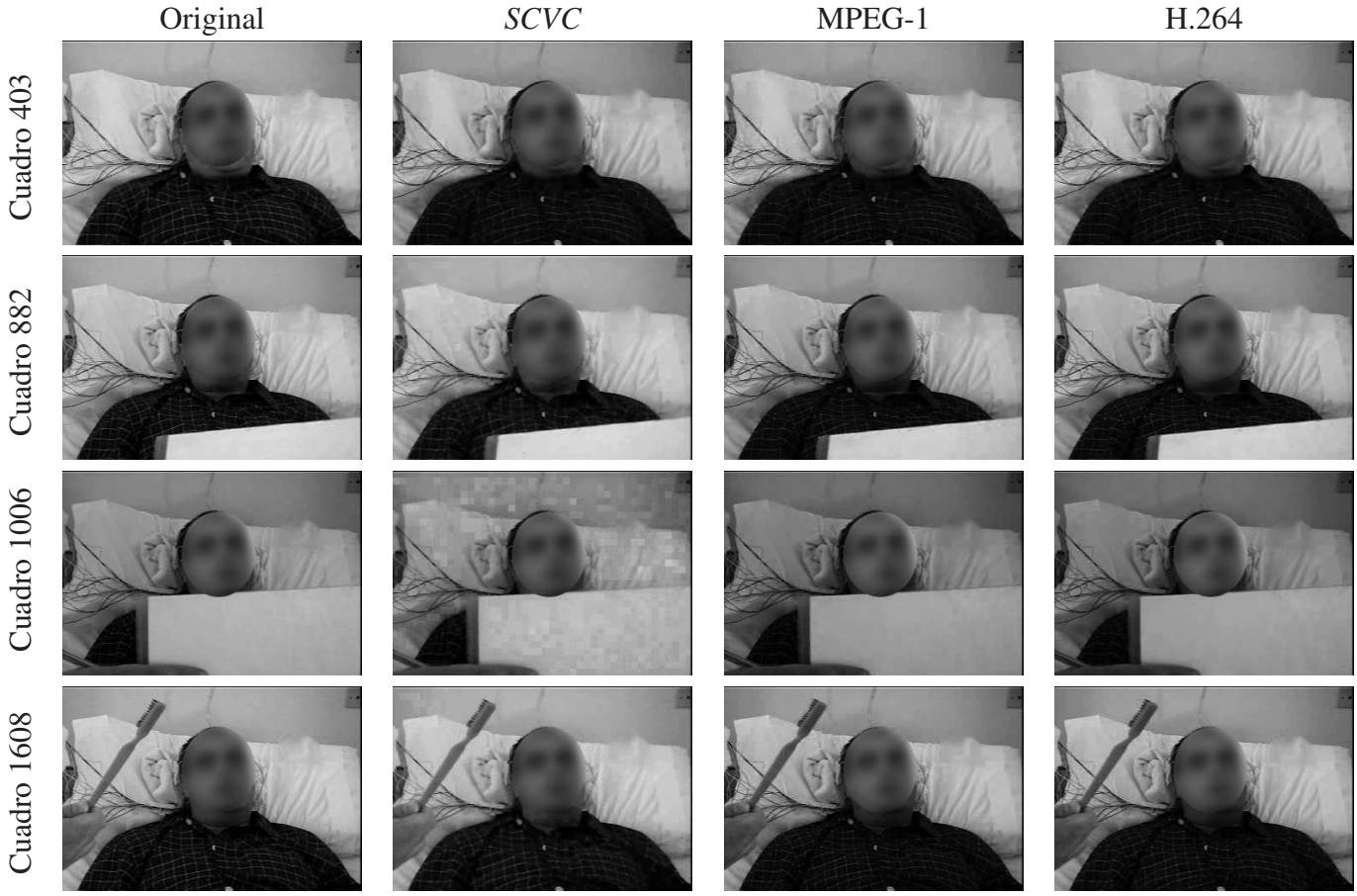


Figura 36: Cuadros 403, 882, 1006 y 1608 de la secuencia *P1_EXT_K1* original y procesado por los tres métodos de compresión analizados.

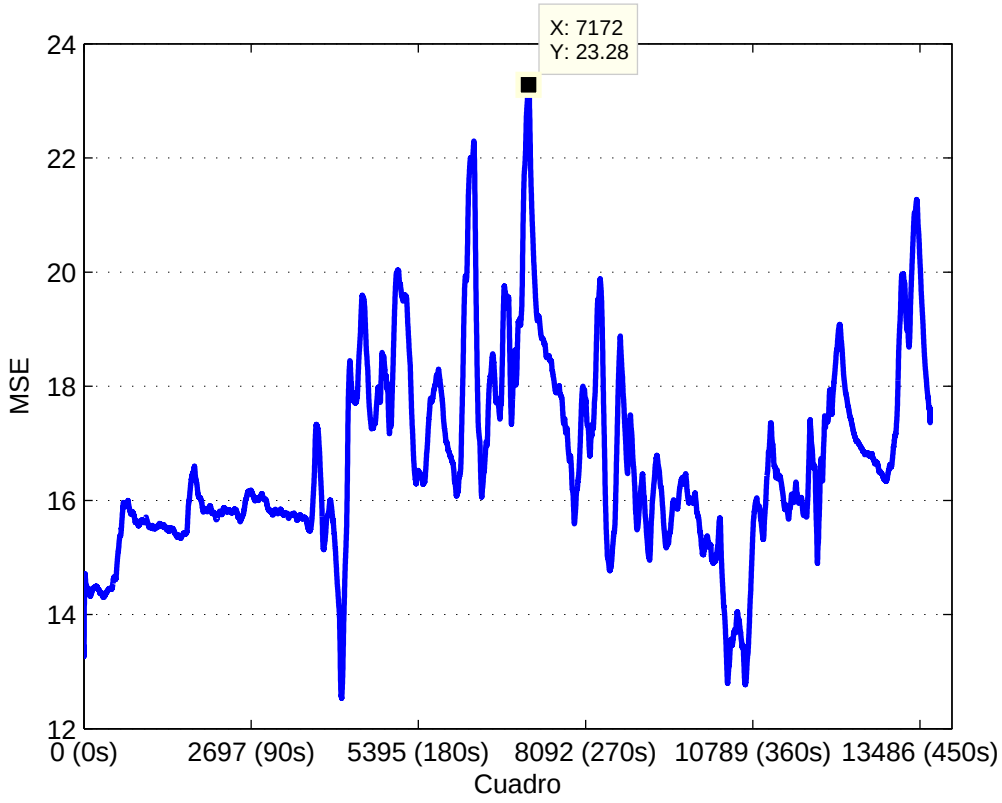


Figura 37: Evolución del *MSE* en función del tiempo para el video *P1_EXT_K4* comprimido con el método presentado en este trabajo.

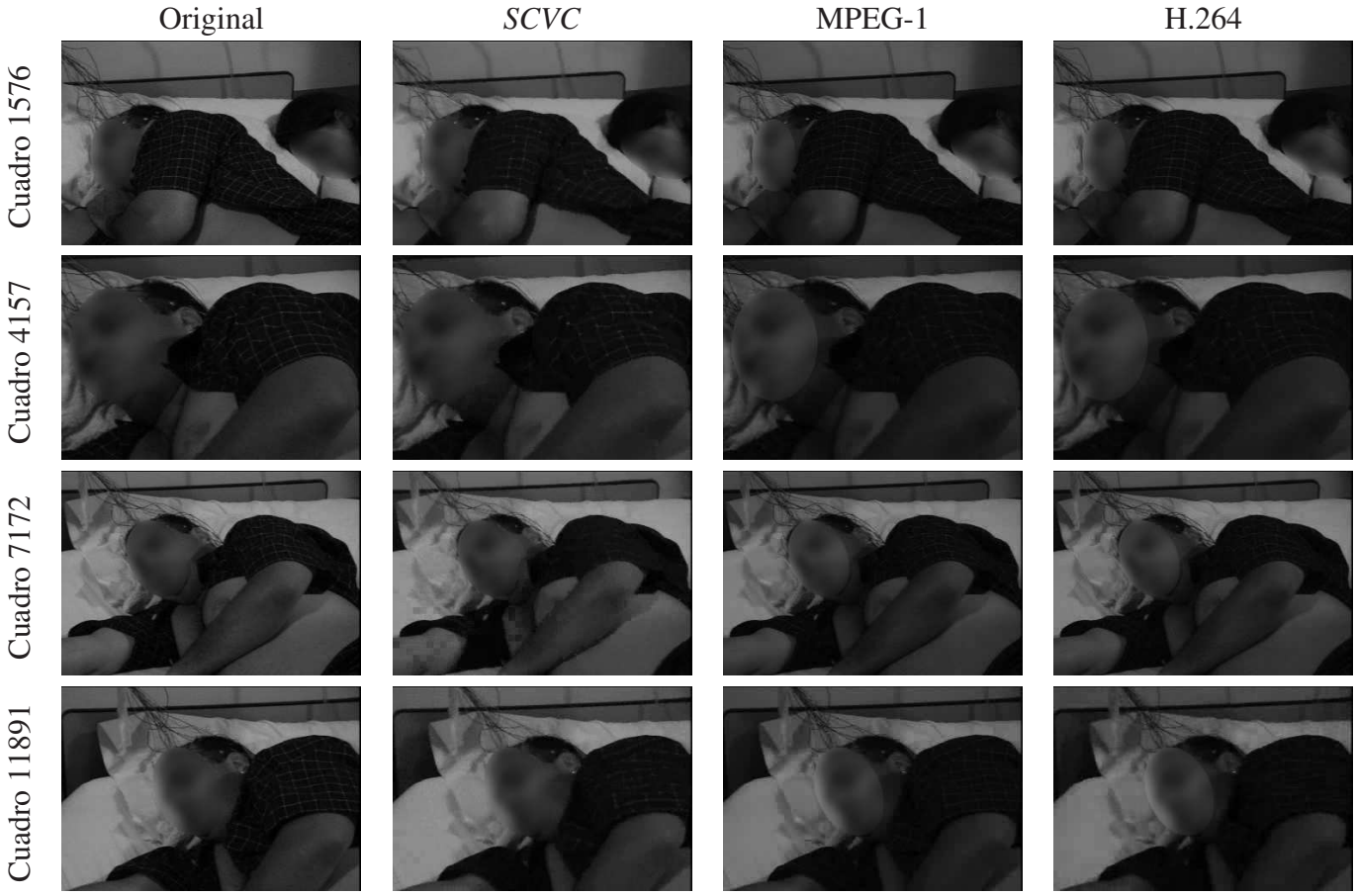


Figura 38: Cuadros 1575, 4157, 7172 y 11891 de la secuencia *P1_EXT_K4* original y procesado por los tres métodos de compresión analizados.

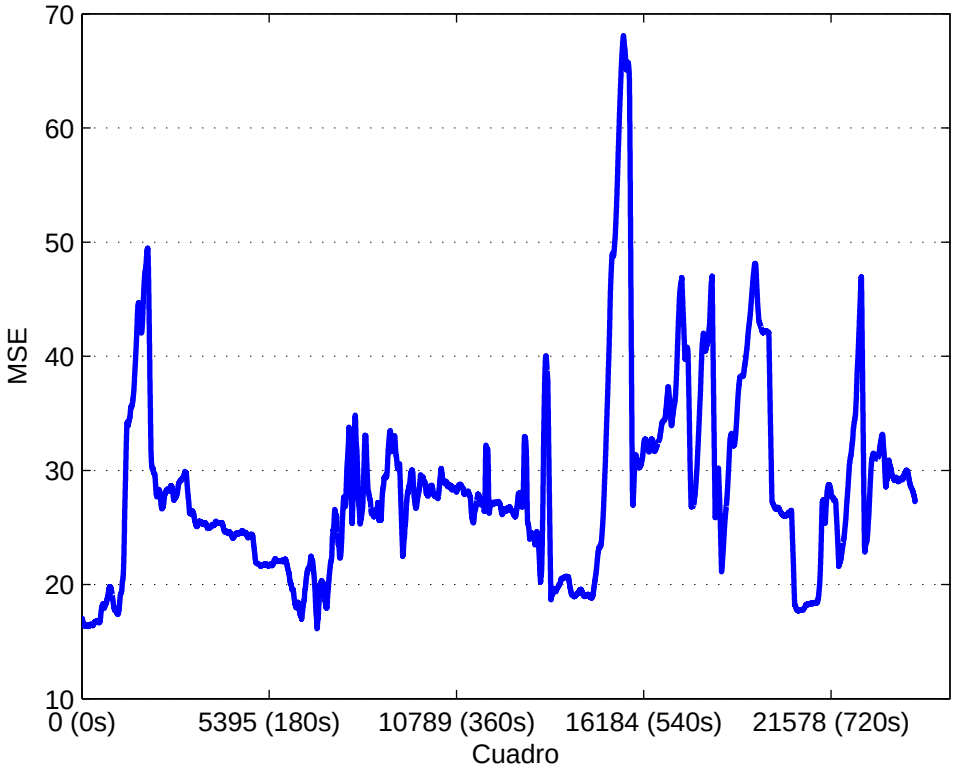


Figura 39: Evolución del *MSE* en función del tiempo para el video *P2_EXT2_K2-3* comprimido con el método presentado en este trabajo.

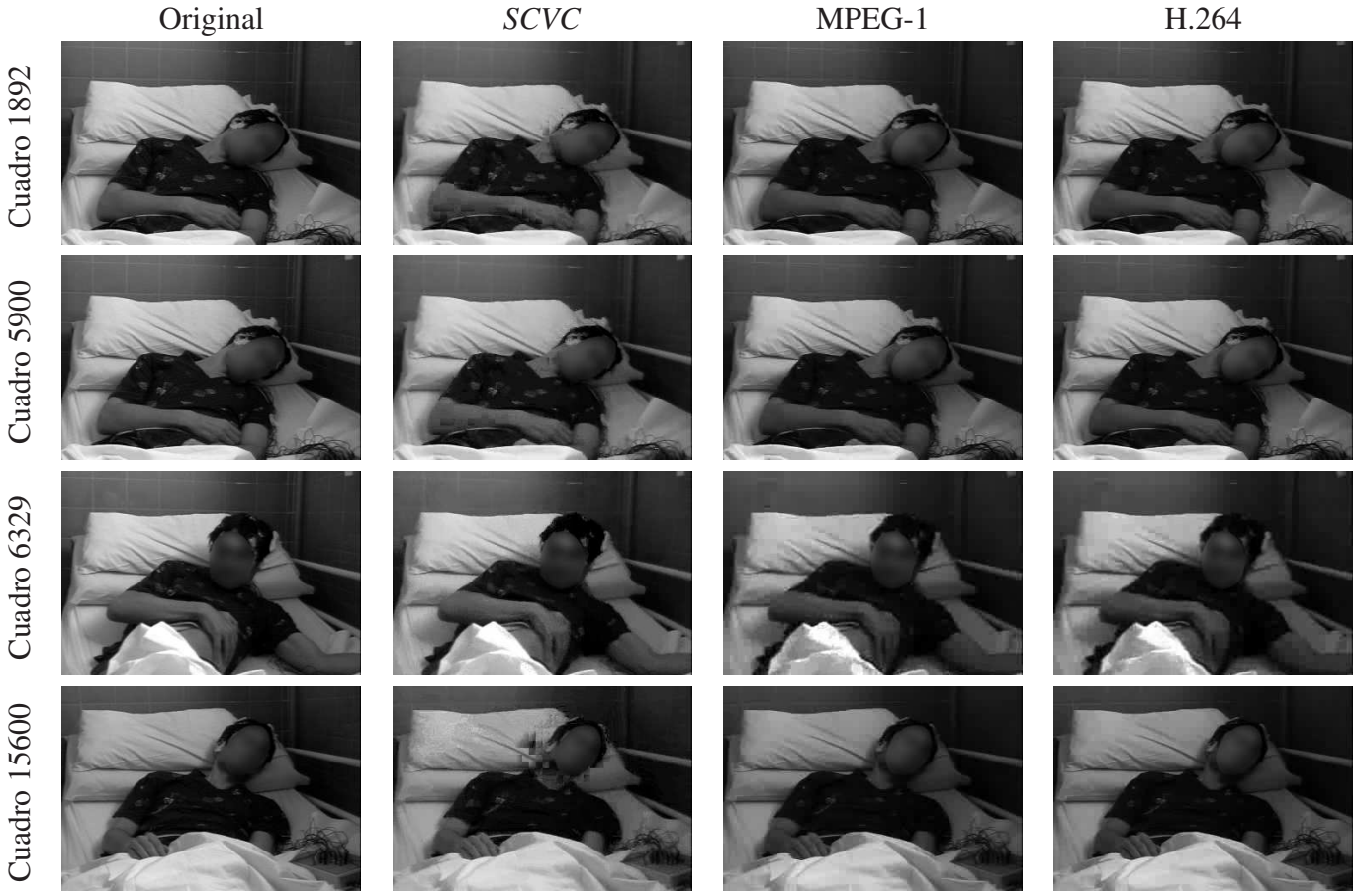


Figura 40: Cuadros 1892, 5900, 6329 y 15600 de la secuencia *P2_EXT2_K2-3* original y procesado por los tres métodos de compresión analizados.

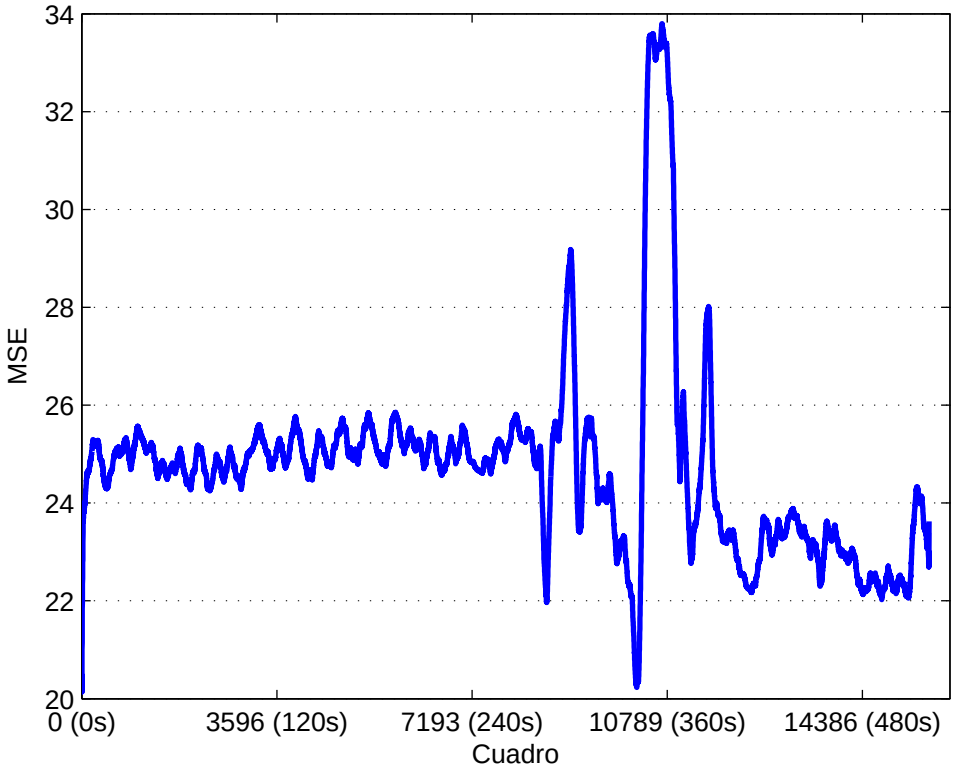


Figura 41: Evolución del *MSE* en función del tiempo para el video *P3_EXT10_K4* comprimido con el método presentado en este trabajo.

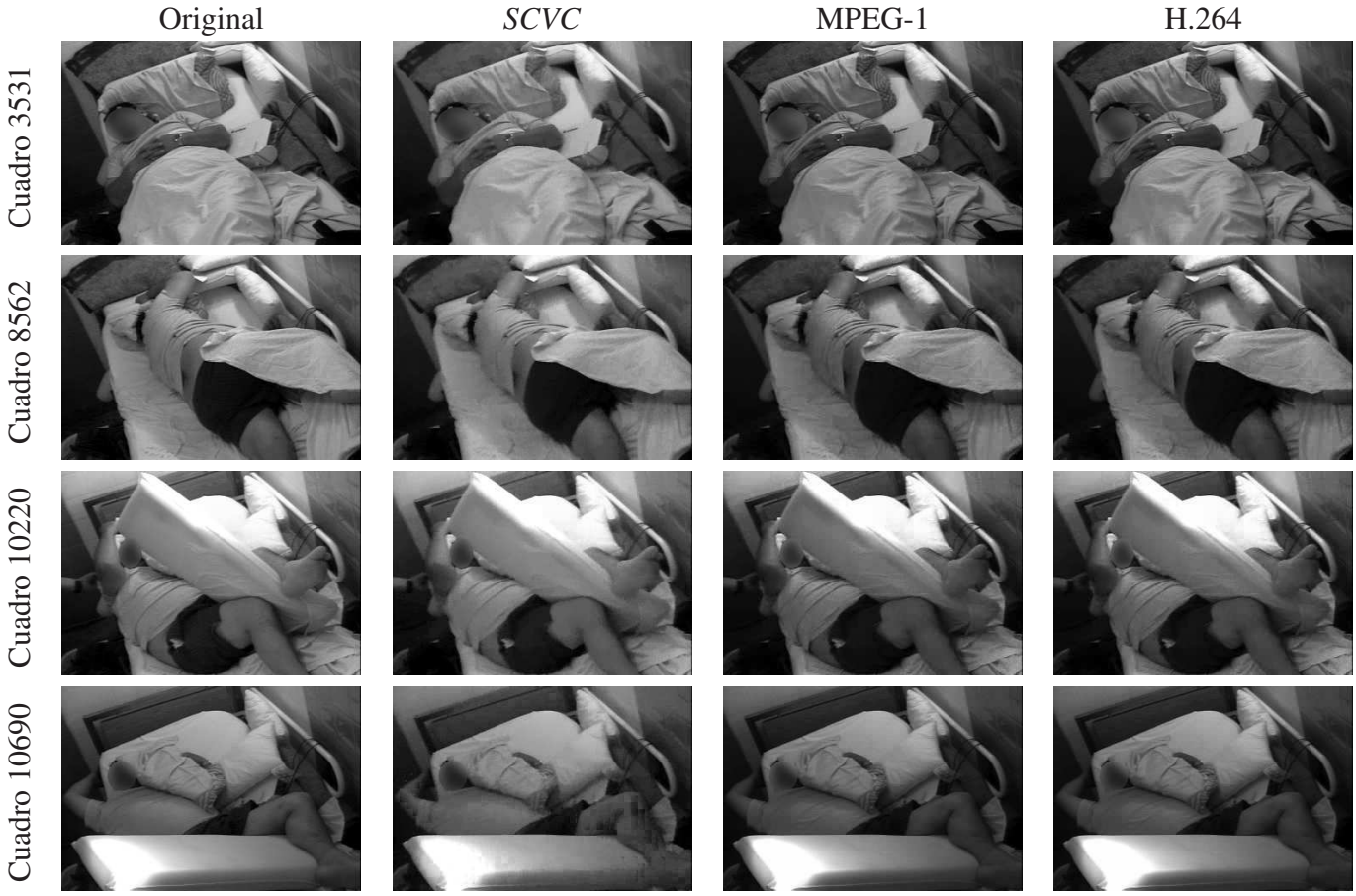


Figura 42: Cuadros 1892, 5900, 6329 y 15600 de la secuencia *P3_EXT10_K4* original y procesado por los tres métodos de compresión analizados.

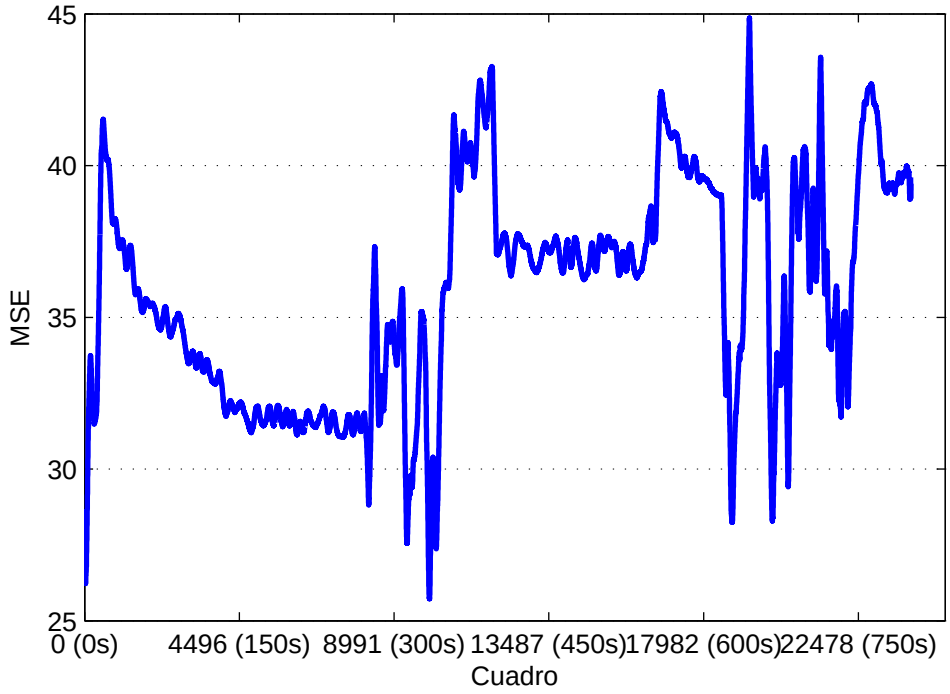


Figura 43: Evolución del *MSE* en función del tiempo para el video *P3_EXT8_K2-3* comprimido con el método presentado en este trabajo.

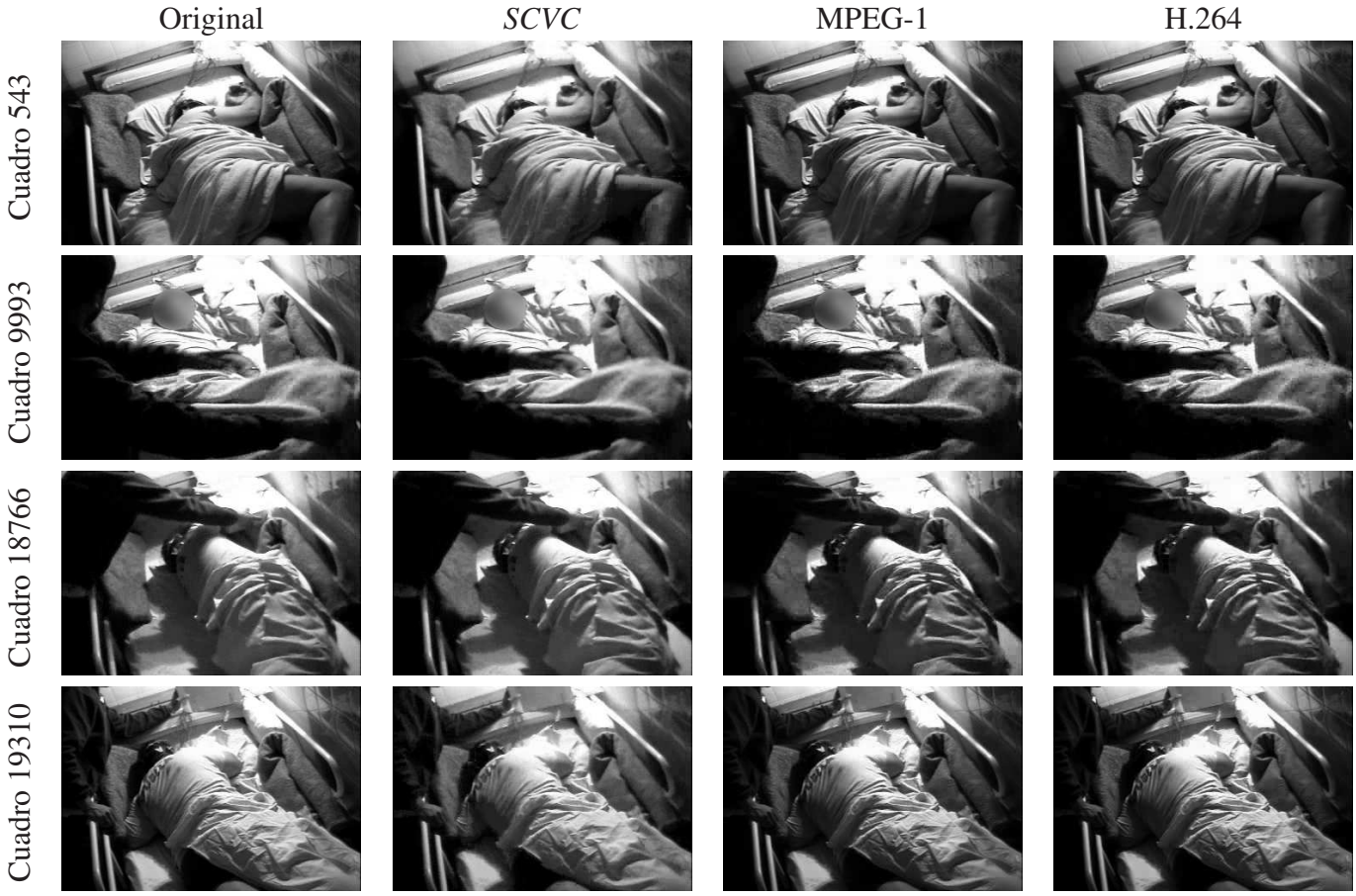


Figura 44: Cuadros 543, 9993, 18766 y 19310 de la secuencia *P3_EXT8_K2-3* original y procesado por los tres métodos de compresión analizados.

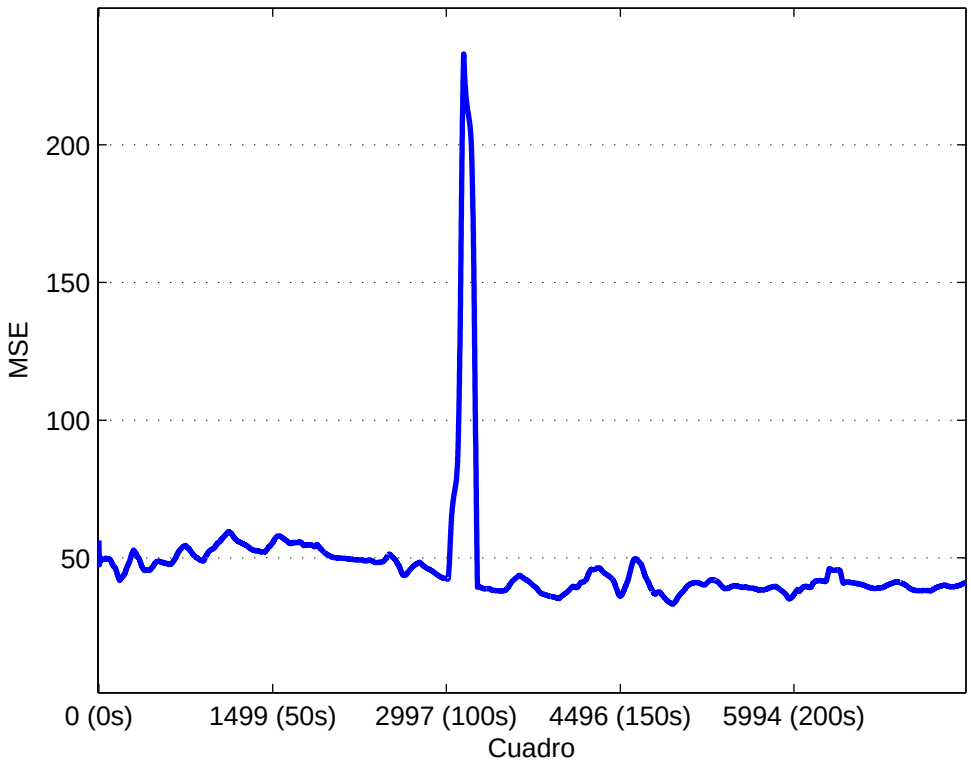


Figura 45: Evolución del *MSE* en función del tiempo para el video *P1_EXT_K3* comprimido con el método presentado en este trabajo.

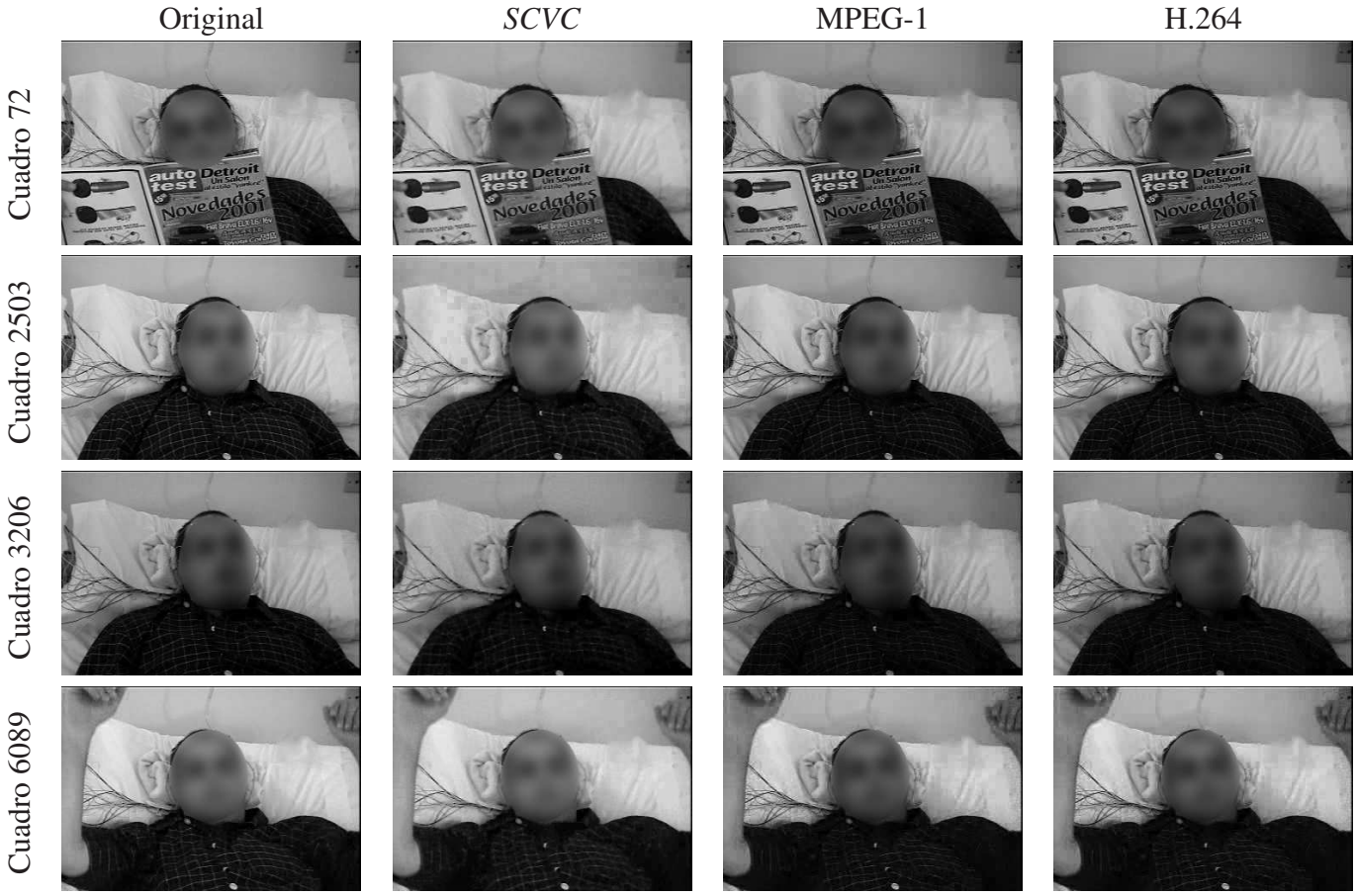


Figura 46: Cuadros 72, 2503, 3206 y 6089 de la secuencia *P1_EXT_K3* original y procesado por los tres métodos de compresión analizados.

C. Detalles de Implementación

El presente trabajo fue realizado mayormente utilizando las siguientes herramientas y lenguajes de programación:

- Matlab Versión V7.2 (R2006a) © 1994-2006 The MathWorks, Inc
- Simulink Versión V6.4 (R2006a) © 1994-2006 The MathWorks, Inc
- Java(TM) SE 1.6 (build 1.6.0_13-b03) © 1994-2009 ORACLE.

Matlab y Simulink forman parte de un mismo paquete de herramientas y se utilizan conjuntamente para prototipar algoritmos y sistemas complejos. El uso de Simulink permite manejar gráficamente el flujo de ejecución de los algoritmos y la comunicación entre los distintos módulos que los componen, además provee una gran cantidad de opciones para la entrada-salida de datos, convirtiendo fácilmente videos e imágenes en matrices de valores y viceversa. Los algoritmos más complejos como el *algoritmo de selección de bloques para enviar* fueron programados íntegramente en Matlab y luego incluidos en los modelos de Simulink. También se utilizó Matlab para analizar por separado cuadros de las secuencias y para generar todos los gráfico incluidos en este trabajo.

La codificación de las secuencias comprimidas a archivos binarios fue programada en Java pues en este lenguaje es más sencillo el manejo de información binaria, además hay una gran cantidad de herramientas y librerías disponibles para codificar secuencias de datos de distintas maneras (en particular se utilizó la librería de codificación aritmética mencionada en [5]). Las clases compiladas en Java fueron incluidas también en el modelo de Simulink de modo que fue posible controlar todo el proceso de codificación/decodificación gráficamente desde la interfaz de dicha herramienta.

Todo el proceso de compresión, desde la lectura del la secuencia de origen en formato AVI, hasta la escritura de la secuencia comprimida en formato binario, es manejado por un único modelo de Simulink. El mismo está compuesto por bloques incluidos en diversos paquetes de Matlab (estos bloques serán detallados a medida que se los mencione), bloques propios programados en Matlab o en Java y submodelos de Simulink compuestos por otros bloques y modelos.

Para este trabajo las secuencias fueron representadas como una sucesión de matrices reales, donde cada matriz corresponde a un cuadro. Los cuadros son leídos, procesados y escritos uno a uno. Simulink es el encargado de sincronizar el proceso y hacer fluir la información de bloque a bloque.

En las próximas secciones se detalla cada uno de estos modelos y los elementos que los componen, así como también los parámetros con los que fueron configurados.

C.1. Modelo principal

El modelo principal (ver Figura 47) contiene a todos los bloques y sub-modelos que implementan el método de compresión de video presentado en este trabajo.

Los primeros tres bloques realizan el preprocesamiento del vido: El video ingresa por el *Puerto 1* etiquetado como "Video In" y luego es procesado por el bloque "Wiener Filter", el cual aplica el filtro de Wiener; el video filtrado alimenta al bloque "Block-Selector Parameters Switcher", el cual se encarga de ajustar los parámetros del *algoritmo de selección de bloques para enviar* en base a la entropía de cada cuadro; después tanto el video filtrado como los umbrales ingresan al bloque "Block Selector", el cual implementa el *algoritmo de selección de bloques para enviar* con parámetros variables.

Una vez procesado por el *algoritmo de selección de bloques para enviar*, un segundo grupo de bloques realiza la detección de cambios: el bloque etiquetado como "Mask Generator" genera un mapa de bits indicando que bloques de cada cuadro presentan alguna diferencia respecto al anterior y qué bloques no; luego el mapa de bits pasa por el bloque "Keyframe Switcher" el cual se encarga de generar un bitmap cuyos valores son todos unos cada 50 cuadros.

Tanto el bitmap de cambios como el video procesado por el *algoritmo de selección de bloques para enviar* ingresan finalmente en un tercer grupo de bloques los cuales realizan las tareas de compresión, descompresión y de escritura de los resultados en formato binario.

A continuación se detalla cada uno de estos bloques.

C.2. Bloque: Block-Selector Parameters Switcher

La función de este bloque es modificar los parámetros del *algoritmo de selección de bloques para enviar* en base a la entropía de cada cuadro del video. La Figura 48 muestra su diagrama interno.

Este bloque funciona de la siguiente manera: la señal ingresa por el puerto de entrada 1 (etiquetado como "Signal"); el bloque "Delay" desfasa en un cuadro la señal y mediante un comparador se obtienen las diferencias entre el cuadro procesado y el anterior; estas diferencias ingresan en el bloque "Entropy Values", el cual calcula la entropía de dichas diferencias; una vez calculada la entropía se utilizan los bloques "Smooth Signal" y "Mean" para suavizar las variaciones entre cuadros consecutivos y para calcular la media acumulada respectivamente; ambos valores ingresan luego al bloque "Block-Selector Threshold Selector" el cual clasifica la entropía del cuadro como *alta*, *media* o *baja* midiendo la distancia entre la entropía media y la entropía del cuadro actual y comparando esta distancia con los umbrales e_L y e_H; finalmente los dos selectores llamados "Switch Multiport" seleccionan los valores de h_μ y h_σ cuyos valores son colocados en los

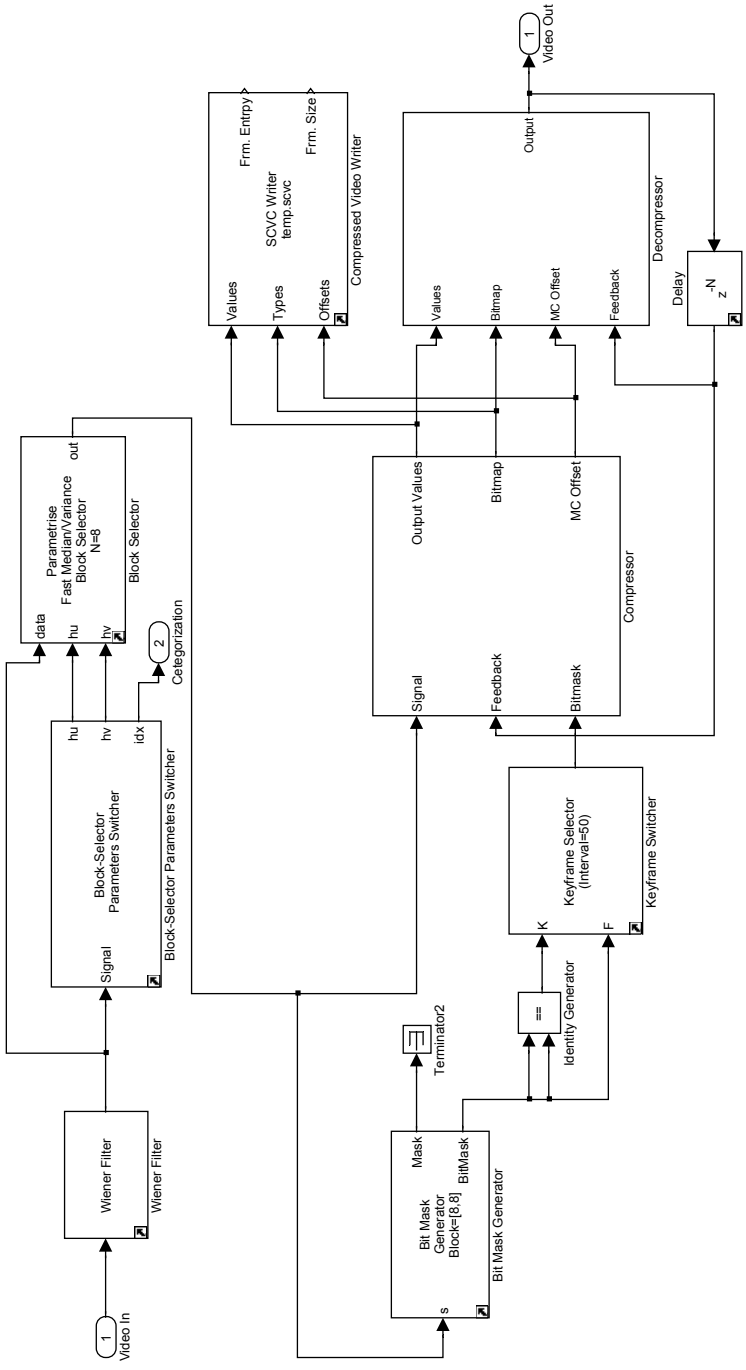


Figura 47: Modelo principal de Simulink.

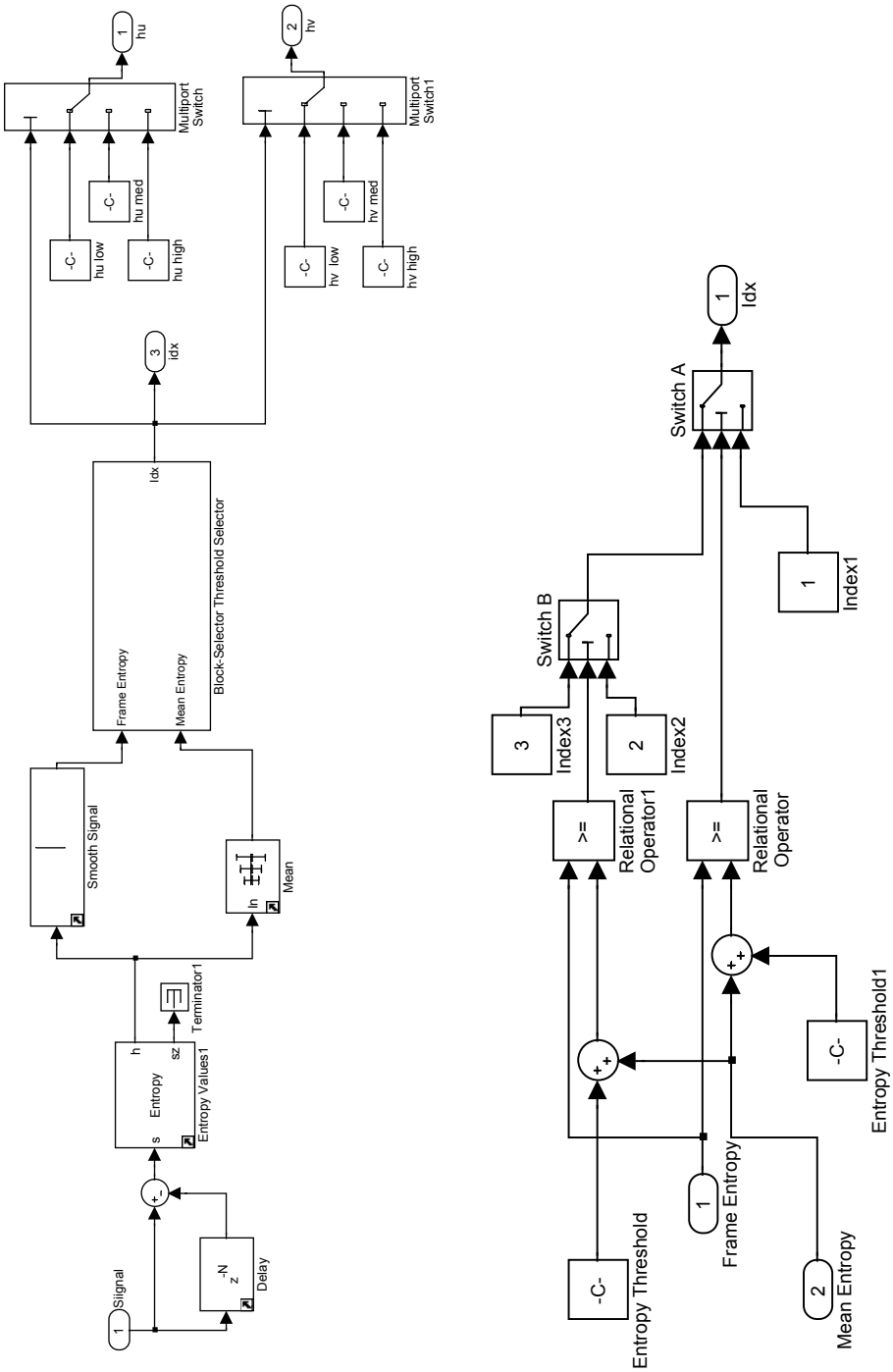


Figura 48: Izquierda: Diagrama interno del bloque “Block-Selector Parameters Switcher”. Derecha: diagrama del sub-bloque “Block-Selector Threshold Selector”.

puertos de salida “hu” y “hv” respectivamente.

Este bloque se configura mediante 9 parámetros, tal como se muestra en la Figura 49.

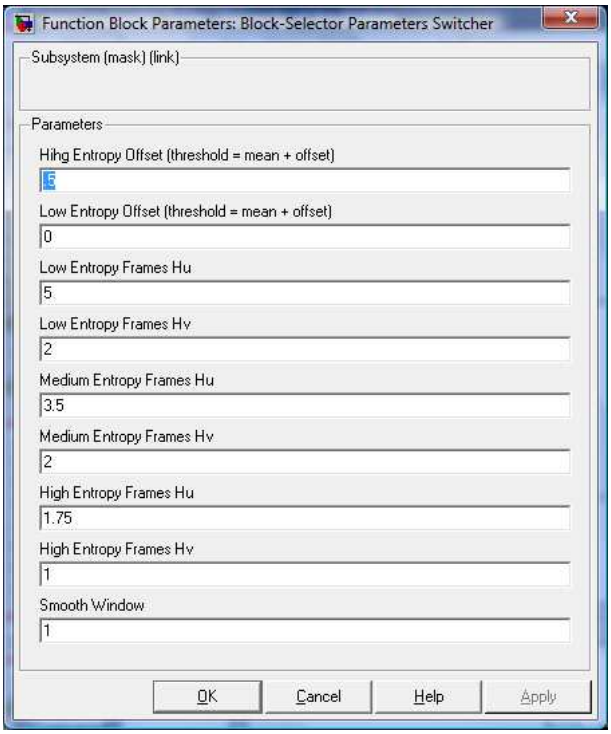


Figura 49: Diálogo de configuración del bloque “Block-Selector Parameters Switcher”.

C.3. Bloque: Block Selector

Este bloque implementa el *algoritmo de selección de bloques para enviar* presentado en la sección 6. A continuación de lista el código Matlab que implementa dicho algoritmo.

Listing 1: Código en Matlab correspondiente al bloque rotulado como “Mean-Variance Noise Reduction”.

```
1 function parametrised_fast_time_mean_variance_blocks_selector(block)
2 % Level-2 M file S-Function for Simulink.
3 %
4 % Time Mean/Deviation Analysis Based Blocks Selector
5 %
6 % Este Simulink Block implementa un mecanismo basado en el análisis de la
7 % media y la varianza de un stream de frames de video
8 % para la detección y eliminación de bloques que no aportan información
9 % relevante a la secuencia de video.
```

```
%
% Parametros de configuración
%
% 1 - N  Tamaño de los bloques
%
% Parametros Variables
%
% 1 - hu Umbral para la media
% 2 - hv Umbral para la varianza
%
% Funcionamiento
%
% Este bloque realiza un análisis por blque bloques de N x N analizando la
% diferencia entre el frame entrante y el promedio de los frames
% anteriores.
%
% Si la diferencia entre las medias del nuevo frame y del promedio
% almacenado supera el umbral hu, el promedio almacenado es descartado y se
% utiliza el nuevo bloque.
%
% Si varianza de la diferencia entre el nuevo bloque y el promedio
% almacenado supera el umbral hv, tambien se descarta el bloque en memoria
% y se utiliza el nuevo bloque.
%
% Si el bloque en memoria no fue descartado, se interpreta que el nuevo
% cuadro es similar al anterior. En este caso, se actualiza la memoria con
% el nuevo bloque y se produce como salida el primer bloque almacenado en
% memoria.

    setup(block);

%endfunction

function setup(block)

  %% Register number of input and output ports
  block.NumInputPorts  = 3;
  block.NumOutputPorts = 1;

  %% Setup functional port properties to dynamically
  %% inherited.
  block.SetPreCompInpPortInfoToDynamic;
  block.SetPreCompOutPortInfoToDynamic;

  %block.InputPort(1).DirectFeedthrough = true;

  %% Set block sample time to inherited
  block.SampleTimes = [-1 0];

  %% Run accelerator on TLC
  block.SetAccelRunOnTLC(true);

  % parametros de entrada
  block.NumDialogPrms     = 2;
  block.DialogPrmsTunable = {'Tunable', 'Tunable'};

  %% Register methods
  block.RegBlockMethod('Outputs',                 @Output);
  block.RegBlockMethod('PostPropagationSetup',    @DoPostPropSetup);
  block.RegBlockMethod('InitializeConditions',    @InitConditions);
  block.RegBlockMethod('Terminate',               @Terminate);
  block.RegBlockMethod('Update',                  @Update);
```

```
72    block.RegBlockMethod('CheckParameters',          @CheckPrms);
73    block.RegBlockMethod('SetInputPortDimensions', @SetInpPortDims);
74    block.RegBlockMethod('SetOutputPortDimensions', @SetOutPortDims);
75
76  %endfunction
77  function CheckPrms(block)
78  %endfunction
79
80  function SetInpPortDims(block, idx, di)
81      block.InputPort(idx).Dimensions = di;
82      if idx == 1
83          block.OutputPort(idx).Dimensions = di;
84      end
85  %endfunction
86
87  function SetOutPortDims(block, idx, di)
88     block.OutputPort(idx).Dimensions = di;
89  %endfunction
90
91  function DoPostPropSetup(block)
92    global mem;
93    mem = {};
94
95    block.NumDworks = 1;
96
97    block.Dwork(1).Name            = 'x1';
98    block.Dwork(1).Dimensions      = 1;
99    block.Dwork(1).DatatypeID      = 0;       % double
100   block.Dwork(1).Complexity      = 'Real';  % real
101   block.Dwork(1).UsedAsDiscState = true;
102
103 %endfunction
104
105 function InitConditions(block)
106     global mem;
107     idx = size(mem,2) + 1;
108     mem{idx} = {};
109     block.Dwork(1).Data = idx;
110
111 %endfunction
112
113 function Output(block)
114     global mem;
115     idx = block.Dwork(1).Data;
116
117     % Parámetros del bloque
118     b = block.DialogPrm(1).Data;     % Tamaño de los bloques
119     mode = block.DialogPrm(2).Data;  % Modo
120
121     % Extiende circularmente el frame para que su tamaño sea múltiplo del
122     % tamaño del bloque.
123     frame = block.InputPort(1).Data;  % Cuadro
124     hu = block.InputPort(2).Data;     % Umbral Media
125     hv = block.InputPort(3).Data;     % Umbral Varianza
126
127     [zf zc] = size(frame);
128
129     difc = b - mod(zc, b);
130     diff = b - mod(zf, b);
131
132     if difc ≠ b
133         frame = [frame frame( :, 1:difc ) ];
```

```
134         bcs = floor(zc / b) + 1;
135     else
136         bcs = floor(zc / b);
137     end
138
139     if diff ≠ b
140         frame = [frame ; frame( 1:diff, : )];
141         bfs = floor(zf / b) + 1;
142     else
143         bfs = floor(zf / b);
144     end
145
146     [nzf nzc] = size(frame);
147
148     % buffer de salida
149     output = zeros([nzf nzc]);
150
151     % Itera sobre cada uno de los bloques del frame
152     for bf = 1:bfs
153         for bc = 1:bcs
154             bcidx = ((bc-1)*b+1):(bc*b);
155             bfidx = ((bf-1)*b+1):(bf*b);
156             bloque = frame(bfidx, bcidx);
157
158             % Si la estructura no existe para este bloque, la crea.
159             try
160                 tmp = mem{idx}.blk{bf, bc}.falg;
161             catch
162                 mem{idx}.blk{bf, bc}.data = bloque;
163                 mem{idx}.blk{bf, bc}.last_output = bloque;
164                 mem{idx}.blk{bf, bc}.last_block = bloque;
165                 mem{idx}.blk{bf, bc}.u = 0;
166                 mem{idx}.blk{bf, bc}.v = 0;
167                 mem{idx}.blk{bf, bc}.acc_u = 0;
168                 mem{idx}.blk{bf, bc}.cout = 0;
169                 mem{idx}.blk{bf, bc}.falg = true;
170             end
171
172             % Calcula la media del bloque, la diferencia y la varianza
173             fdif = mem{idx}.blk{bf, bc}.last_block - bloque;
174             u = mean(mean(fdif));
175             v = var(fdif(:));
176             mem{idx}.blk{bf, bc}.u = u;
177             mem{idx}.blk{bf, bc}.v = v;
178
179             % Verifica si los valores obtenidos se encuentran dentro de los
180             % umbrales definidos.
181              if ( mode == 1 ...
182                 && abs(mem{idx}.blk{bf, bc}.acc_u - u) < hu ...
183                 && v ≤ hv ...
184              ) || ( ...
185                 mode == 2 && ...
186                 max(max(abs(mem{idx}.blk{bf, bc}.data - bloque))) < hu...
187                 && v ≤ hv ...
188              )
189
190                 output(bfidx, bcidx) = mem{idx}.blk{bf, bc}.last_output;
191
192                 mem{idx}.blk{bf, bc}.cout = mem{idx}.blk{bf, bc}.cout + 1;
193
194                 mem{idx}.blk{bf, bc}.acc_u = ...
195                     (mem{idx}.blk{bf, bc}.acc_u * ...
```

```
196                     (mem{idx}.blk{bf, bc}.cout - 1) + u) ...
197                     / mem{idx}.blk{bf, bc}.cout;
198
199                 mem{idx}.blk{bf, bc}.data = ...
200                     (mem{idx}.blk{bf, bc}.data ...
201                     * (mem{idx}.blk{bf, bc}.cout - 1) ...
202                     + bloque) / mem{idx}.blk{bf, bc}.cout;
203             else
204                 output(bfidx, bcidx) = bloque;
205                 mem{idx}.blk{bf, bc}.data = bloque;
206                 mem{idx}.blk{bf, bc}.last_output = bloque;
207                 mem{idx}.blk{bf, bc}.acc_u = u;
208                 mem{idx}.blk{bf, bc}.cout = 1;
209             end
210             mem{idx}.blk{bf, bc}.last_block = bloque;
211         end
212     end
213
214     block.OutputPort(1).Data = output( (1:zf), (1:zc) );
215 %endfunction
216
217 function Update(block)
218 %endfunction
219
220 function Terminate(block)
221     global mem;
222     clear mem;
223 %endfunction
```

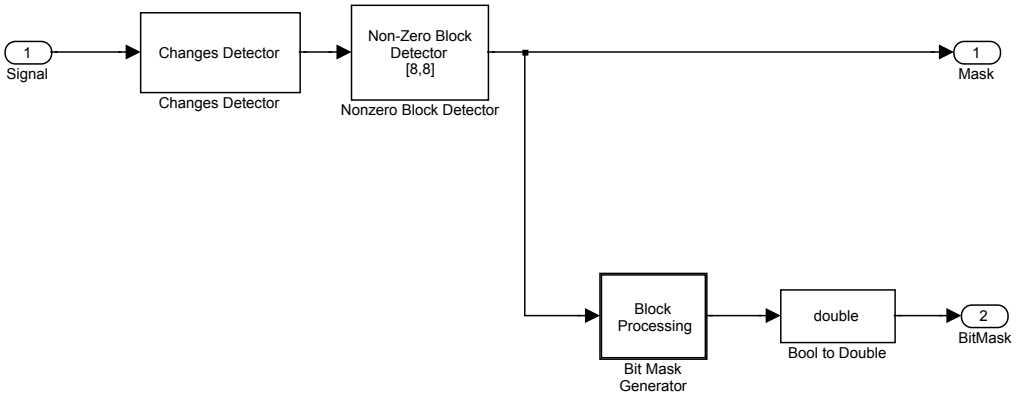
C.4. Bloque: Mask Generator

Este bloque compara cada cuadro con su predecesor para detectar qué bloques presentar cambios. Por cada cuadro de la secuencia genera dos salidas llamadas “Mask” y “BitMask”. “Mask” es una máscara de bits del mismo tamaño que el cuadro original pero donde cada bloque tiene los valores 1 ó 0 dependiendo de si se detectaron cambios en el bloque correspondiente o no. La salida “BitMask” contiene la misma información que la anterior pero organizada en un matriz booleana que contiene un único valor por cada bloque.

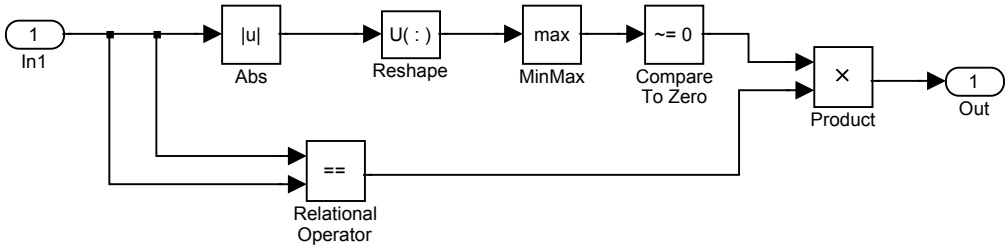
La Figura 50 muestra el diagrama interno de este bloque. El bloque “Changes Detector” es el encargado de comparar el cuadro entrante con su predecesor y genera como salida la diferencia pixel a pixel. Este bloque tiene una memoria de un cuadro de modo que no necesita ningún tipo de retroalimentación.

La salida es luego procesada por el bloque “Nonzero Block Detector” cuya función es detectar qué bloques tienen al menos un valor distinto de cero y generar como salida bloques enteros con valores 1 ó 0 según corresponda.

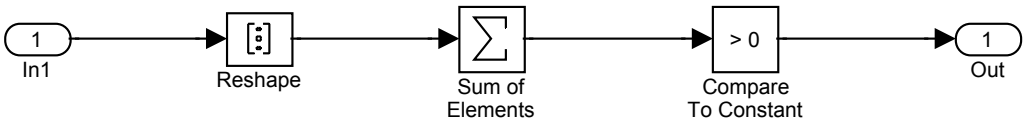
La salida de este bloque es inyectada en la salida “Mask”, a su vez, la misma señal es procesada por el bloque “Bit Mask Generator” que se encarga de generar la matriz booleana con una componente por bloque. La salida de este último bloque es de tipo booleano, de modo que es necesario convertirla un double; de



Mask Generator



Non-Zero Block Detector



Bit Mask Generator

Figura 50: Modelo correspondiente al bloque “Mask Generator” y sus sub-bloques.

esto se encarga el bloque “Bool to Double”, cuya salida es inyectada en el puerto “BitMask”.

C.5. Compressor

Este bloque es el encargado de realizar la *compensación de movimiento* (Sección 7), aplicar la transformada *DCT* y la cuantización (Sección 8.1) y clasificar cada bloque de la señal de salida en las cuatro clases presentadas en la Tabla 12. La Figura51 muestra el diagrama correspondiente a este bloque.

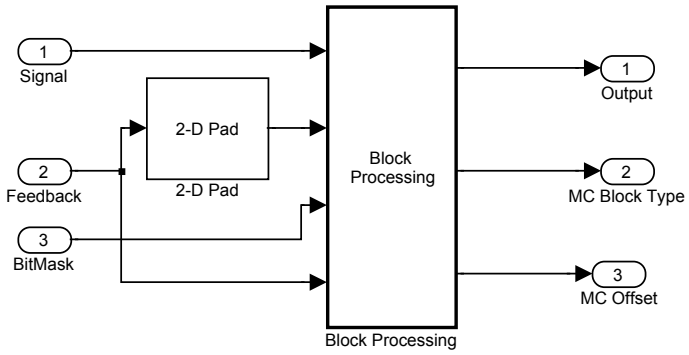


Figura 51: Diagrama interno del bloque “Compressor”.

La interfaz de configuración de este bloque (ver Figura 52) permite ajustar los parámetros h_L y h_H.

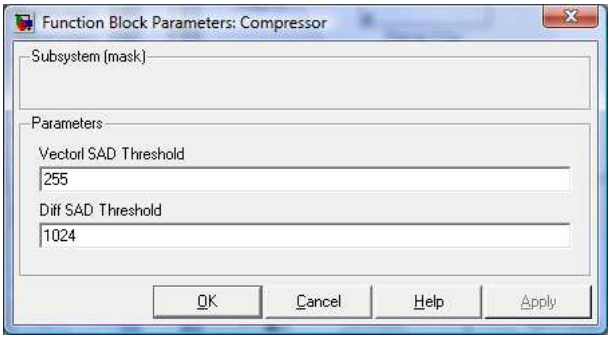


Figura 52: Diálogo de configuración del bloque “Compressor”.

Cada bloque del cuadro es procesado en forma independiente, la herramienta “Block Processing” incluida como parte de la librería *Video and Image Processing Blockset* de Simulink es la encargada de tomar las señales, dividirlas en

bloques independientes y procesar cada bloque por separado mediante un sub-modelo definido por el usuario. Cada salida de dicho sub-modelo es combinada nuevamente de modo de genera un collage que une todos los bloques procesados.

En este caso el bloque "Block Processing" es alimentado por 4 señales:

- La señal previamente procesada por el bloque "Block Selector" recibida a través del puerto "Signal". Esta señal es dividida en bloques de 8×8 píxeles.
- La segunda señal corresponde al cuadro anterior de la secuencia luego de aplicarle un *padding* que, por cada bloque de 8×8, genera un *superbloque* de 24×24 que es utilizado como área de búsqueda en el proceso de compensación de movimiento. Esta operación la realiza el bloque "2-D Pad" que también forma parte de la librería *Video and Image Processing Blockset* de Simulink.
- La tercera entrada recibe la máscara de bits que indica qué bloques presentaron cambios con respecto al cuadro anterior. Estos valores son procesados de a uno por vez.
- Y por último la recuperada del cuadro anterior. Esta señal también es dividida en bloques de 8×8 píxeles.

La figura 53 muestra el submodelo utilizado por la herramienta "Block Processing" para procesar cada bloque del video, éste recibe como entrada las cuatro señales de a un bloque por vez.

El funcionamiento de este modelo es el siguiente: en primer lugar, el bloque correspondiente al cuadro anterior (que ingresa por el puerto "In2") y el superbloque que ingresa por el puerto "In1" alimentan el bloque "SAD1" que es en encargado de realizar el proceso de compensación de movimiento.

El bloque "SAD" (por la sigla en inglés de *Sum of Absolute Diferences* or *Suma de Diferencias Absolutas*) también forma parte de la librería *Video and Image Processing Blockset* de Simulink y funciona calculando la suma de diferencias absolutas entre la señal que ingresa por el puerto "I" (en este caso un bloque del cuadro actual) contra todas las posibles posiciones de la imagen que ingresa por el puerto "Template", que en este caso es el *superbloque* correspondiente al cuadro anterior. Como contraparte genera tres salidas:

- Por el puerto "idx" devuelven un array de dos posiciones conteniendo el *offset* para el cual la suma fue mínima.
- Por el puerto "NVals" devuelve la mínima suma de diferencias absolutas.
- La salida del puerto "NValid" no es utilizada en este trabajo.

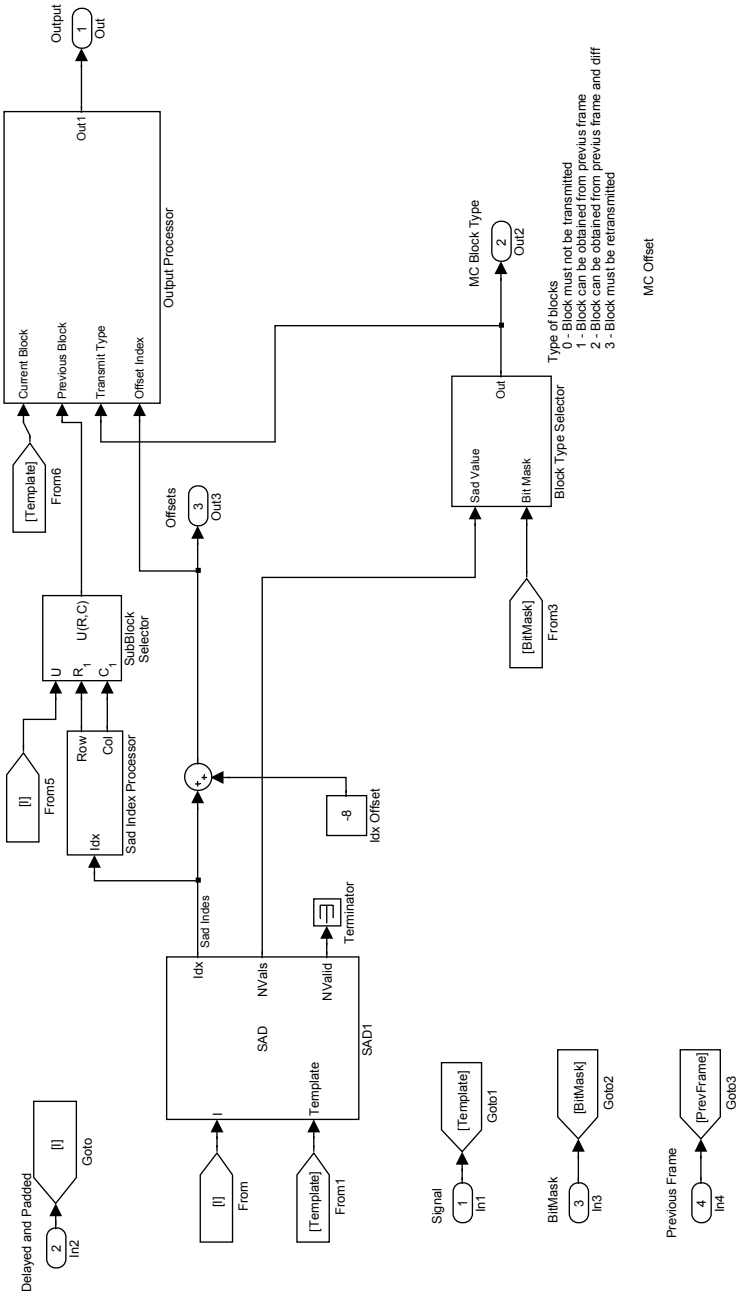


Figura 53: Diagrama interno del sub-modelo utilizado para procesar cada bloque del video dentro del bloque “Compressor”

.

Luego el bloque “Sad Index Processor” divide la señal “idx” en sus componentes “row” y “col” para utilizarlas como índices en el bloque “SubBlock Selecor” el cual extrae del *superbloque* el sub bloque de 8×8 sobre el cual se halló la mínima suma de diferencias absolutos. El bloque resultante es devuelto en el puerto “$U(R, C)$”.

Paralelamente, en el bloque “Block Type Selector” la señal “NVals” es comparada con los umbrales h_L y h_H (presentados en la sección 7) y con el valor del bitmap correspondiente al bloque que se está procesando, el cual ingresa por el puerto “In3”. La función de este sub-modelo es clasificar al bloque procesado según las clases presentadas en la Tabla 12. La clasificación obtenida (representada como valores enteros en el rango $[0, 3]$) alimenta la salida “Bitmap” del bloque “Compressor”.

Por último, el bloque del cuadro actual, el bloque del cuadro anterior obtenido por compensación de movimiento, la clasificación del bloque y el *offset* alimentan al bloque “Output Processor” que es el responsable de generar la señal de salida “Output Values” del bloque “Compressor”. El sub-modelo que implementa este bloque puede verse en la Figura 54.

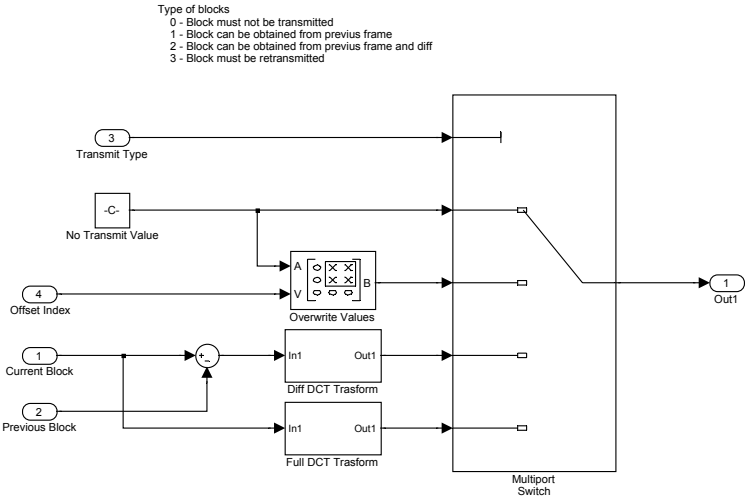


Figura 54: Diagrama interno del sub-modelo “Output Processor” responsable de alimentar la salida “Output Values” del bloque “Compressor”

Este sub-modelo se basa en el bloque “Multiport Switch” que funciona básicamente como un conmutador que coloca en la salida el contenido de uno de los puertos de entrada en base al valor que entra por el primer puerto llamado *Puerto de Control*. El valor recibido por el puerto control no es otro sino la clasificación del bloque. Las posibles salidas de este bloque son:

- Si el *Puerto de Control* recibe el valor 0: la salida del bloque será una matriz

de 8×8 ceros.

- Si el *Puerto de Control* recibe el valor 1: la salida será simplemente el offset obtenido del bloque "SAD1".
- Si el *Puerto de Control* recibe el valor 2: la salida del bloque será la diferencia entre el sub bloque de 8×8 del *superbloque* para el cual la suma de diferencias absolutas fue mínimo y el bloque del cuadro actual, transformada y cuantizada tal como se explica en la Sección 8.1.
- Si el *Puerto de Control* recibe el valor 3: la salida del bloque será el bloque actual transformado y cuantizado tal como se explica en la Sección 8.1.

C.6. Decompressor

Este bloque realiza la tarea inversa al anterior: recibe las señales "Values", "Bitamp", "MC Offset" y el cuadro anterior recuperado (que ingresa por el puerto "Feedback") y reconstruye el cuadro actual. Este bloque es, a su vez, es el responsable de generar el *feedback* que es utilizado para comprimir el cuadro siguiente.

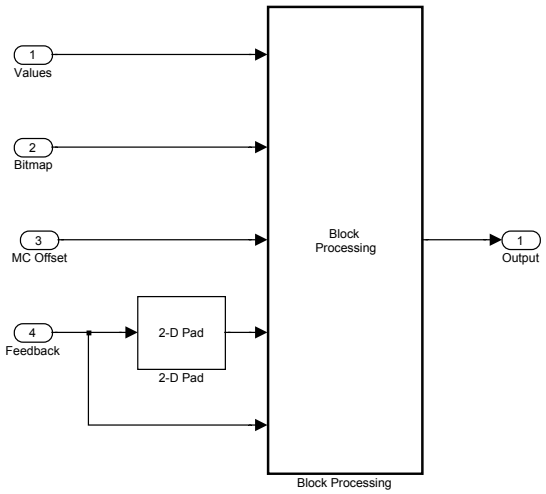


Figura 55: Diagrama interno del bloque "Decompressor".

La figura 55 muestra el diagrama interno de este bloque. El tratamiento que este bloque da a las señales es similar, en cierto modo, al bloque "Compressor", con la diferencia que en este caso el *padding* se aplica sobre el *feedback*. La entrada "Values" se procesa en bloques de 8×8, los valores de "Bitamp" se procesan de a uno (como escalares) y los de "MC Offset" de a pares. La señal "Feedback"

ingresa al “Block Processing” (luego del padding) como *superbloques* y como bloques de 8×8 píxeles.

La Figure 56 muestra el sub-modelo encargado de procesar cada uno de los bloques del cuadro a descomprimir. La salida generada por este sub-modelo para cada bloque es combinada en una única matriz que resulta en el cuadro recuperado.

A simple vista se puede apreciar que este submodelo es mucho menos complejo que el anterior, esto se debe a que el proceso de descompresión es mucho más simple que el de compresión. Se pueden distinguir tres secciones bien definidas.

La primera, conformada por el bloque “I-DCT Transform” (cuya implementación se puede ver en la Figura 57) el cual, en primer lugar, invierte la cuantización de la señal multiplicándola por 16 si el bloque es de *Clase 2* o por la matriz de cuantización *P* para bloques de *Clase 3*. Luego le aplica la antitransformada DCT y finalmente, sólo para los bloques de *Clase 3*, suma 128 a cada elemento.

La segunda, conformada por los bloques “Idx Offset”, “Sad Index Processor”, “SubBlock Selector” y un sumador que extrae del superbloque del cuadro anterior obtiene el bloque de 8×8 apuntado por el *offset*. Y por último el selector “Multipor Switch” genera como salida del submodelo el bloque correspondiente a la clase según el valor recibido en la entrada “In2”.

C.7. Compressed Video Writer

Este bloque implementa los pasos detallados en las secciones 8.2 y 8.3. Consiste básicamente en código Matlab que llama a una librería programada en Java la cual es la responsable de la codificación.

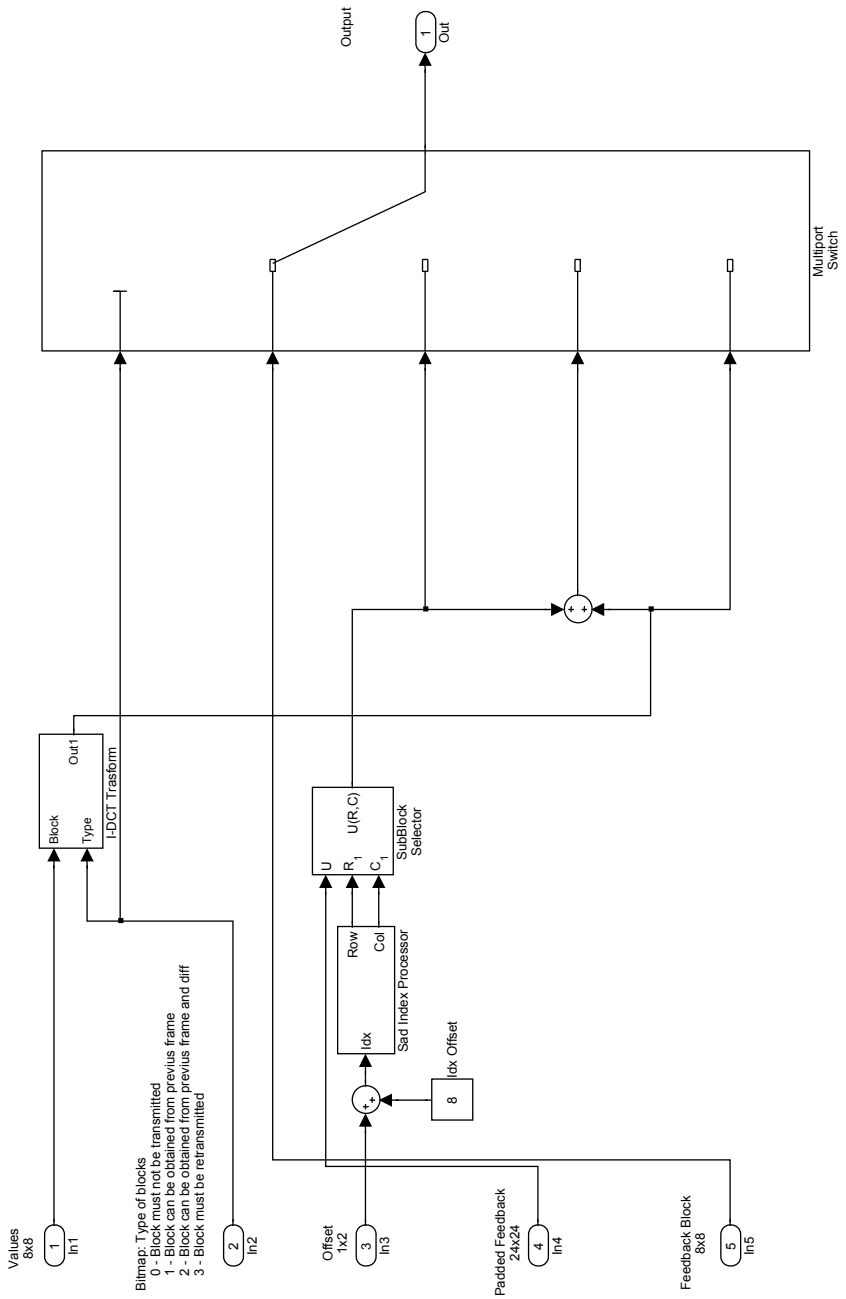


Figura 56: Diagrama interno del sub-modelo “Block Processing” dentro del bloque “Decompressor”.

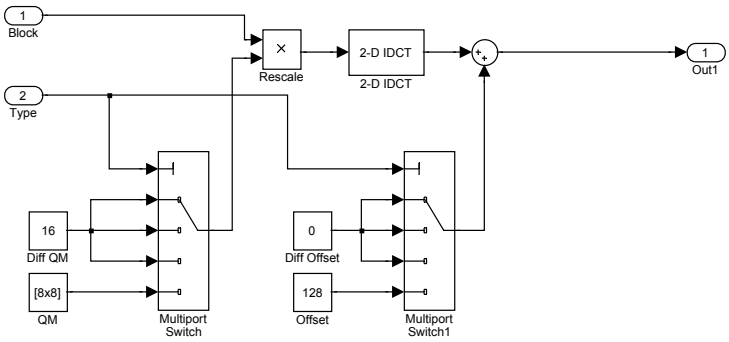


Figura 57: Diagrama interno del bloque “I-DCT Transform”.

D. Descompresión de secuencias

El proceso de descompresión de una secuencia de video es mucho más directo que la compresión. Los siguiente pasos permiten descomprimir una secuencia de video a partir de un archivo con extensión *.scvc* generado a partir del algoritmo presentado en este trabajo.

1. Partiendo del archivo binario con extensión *.scvc* (descripto en la Sección 8.3), se obtiene la codificación intermedia (Sección 8.2) mediante un decodificador aritmético con idénticas características al utilizado para codificar la secuencia.

2. Una vez obtenida la codificación intermedia, se decodifica para obtener el *bitmap* de bloques, los *offsets* y las matrices cuantizadas para correspondientes a los bloques de clases 2 y 3.

3. Se invierte la cuantización de los bloques de *Clase 2* multiplicando sus valores por 16 y la de los bloques de *Clase 3* multiplicando componente a componente cada bloque con la matriz de cuantización de la Figura 18. Finalmente se suma 128 a todos píxeles de los bloques de *Clase 3*.

4. Se aplica la *Anti-Transformada Discreta del Coseno* a cada uno de los bloques las clases 2 y 3.

5. Finalmente se reconstruye el cuadro bloque a bloque en base a su clasificación en el *bitmap* de bloques:

   - Para los bloques de *Clase 0* se repite la misma información que la decodificada para el cuadro anterior.
   - Para cada bloque de *Clase 1* se utiliza el desplazamiento correspondiente para obtener los valores del bloque a partir del cuadro anterior.
   - Para los bloques de *Clase 2* se obtienen también los valores a partir del cuadro anterior en base al desplazamiento para luego sumarlos a los valores del bloque de diferencia de bloques correspondientes.
   - Para los bloques de *Clase 3* se utiliza la matriz resultante de la *Anti-Transformada Discreta del Coseno*.

Referencias

[1] Babu, R., and Makur, A. Object-based surveillance video compression using foreground motion compensation. pp. 1–6.

[2] Chiariglione, L. Short mpeg-1 description. *ISO/IEC JTC1/SC29/WG11 - MPEG96* (Jun. 1996).

[3] Clark, R. N.. Digital camera image noise. http://www.cambridgeincolour.com/tutorials/image-noise.htm, 2008.

[4] CONICET, P. La epilepsia sería más habitual de lo que se cree. http://www.universia.com.ar/materia/materia.jsp?materia=25456, Sep. 2007.

[5] Diego Marco López, J. G. M., and Álvarez Álvarez, T. Codec: Aplicación para el aprendizaje de técnicas de codificación y decodificación de datos. http://www.isa.cie.uva.es/proyectos/codec/teoria32.html.

[6] Foi, A. Pointwise shape-adaptive dct image filtering and signal-dependent noise estimation. Tech. Rep. Publication 710, Tampere University of Technology, Dec. 2007. ISBN 978-952-15-1922-2.

[7] Ghanbari, M. Video coding: an introduction to standard codecs. Tech. rep., London: Institution of Electrical Engineers, 1999.

[8] Herbulot, A., Blotz, S., Debreuve, E., and Barlaud, M. Robust motion-based segmentation in video sequences using entropy estimator. *Int. Conf. Image Processing* (2006).

[9] Generic coding of audido-visual objects part 2: Visual. Tech. rep., ISO/IEC 14 496-2, 2001.

[10] Julio Artieda, Jorge Iriarte, C. V. Funcionamiento e indicaciones de una unidad de video-eeg. http://neurologia.rediris.es/congreso-1/conferencias/p-tecnologicas-3.html. Servicio de Neurofisiología Clínica. Unidad de Epilepsia y Sueño. Departamento de Neurología y Neurocirugía. Clínica Universitaria de la Universidad de Navarra, Pamplona.

[11] Koga, T., Iinuma, K., Hirano, A., Iijima, Y., and Ishiguro, T. Motion-compensated interframe coding for video conferencing. *IEEE NTC* (1981), pp. 531–534.

[12] Liu, Q. New change detection models for object-based encoding of patient monitoring video. *School of Engineering of the University of Pittsburgh* (2005).

[13] Liu, Q., Chen, D., Sun, M., and Sclabassi, R. J. Specialized video and physiological data coding system for remote monitoring. http://www.informedia.cs.cmu.edu/documents/icme06_liu.pdf.

[14] Meier, T., and Ngan, K. N.. Automatic segmentation of moving objects for video object plane generation. *Circuits and Systems for Video Technology, IEEE Transactions on 8*, 5 (1998), 525–538.

[15] Navarra, G. Tecnología de punta al alcance de todos. http://www.lanacion.com.ar/nota.asp?nota_id=449760, Nov. 2002.

[16] Poynton, C. *A Technical Introduction to Digital Video*. John Wiley & Sons, 1996.

[17] Rissanen, J., and Langdon, G. Arithmetic coding. *IBM Journal of Research and Development Vol. 23*, No. 2 (Mar. 1979), pp. 149–162.

[18] Salomon, D. *Data Compression: The Complete Reference*, second ed. pub-SV, 2004.

[19] Shyamsunder, R., Eswaran1, C., and Sriraam, N. Compression of patient monitoring video using motion segmentation technique. *Journal of Medical Systems*, No. 31 (2007), pp. 109–116.

[20] Sikora, T. The mpeg-4 video standard verification model. *IEEE Trans. on Circuits and System for Video Technology Vol. 12*, No. 1 (Sep. 1997), pp. 19–31.

[21] Wallace, G. K. The jpeg still picture compression standard. *IEEE Transactions on Consumer Electronics 38*, 1 (Feb 1992), xviii–xxxiv.

[22] Wang, Y., Ostermann, J., and Zhang, Y.-Q. Digital video processing and communications. *Prentice Hall 2002 (printed 2001)* (Jan. 2002).

[23] Wei, J., and nian Li, Z. An efficient two-pass map-mrf algorithm for motion estimation based on mean field thoery. *IEEE Transactions on Image Processing Vol. 9*, No. 6 (Sep. 1999), pp. 960–720.

[24] Wiegand, T., Sullivan, G. J., Bjontegaard, G., and Luthra, A. Overview of the h.264/avc video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology 13*, 7 (July 2003), 560–576.

[25] Wiener, N. *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*. New York: Wiley, 1949. ISBN 0-262-73005-7.

[26] YATES, R. Mean of a white-noise process. *Digital Signal Labs* (Aug. 2009).

[27] ZANG, J. The mean field theory in em procedures for blind markov random field image restoration. *IEEE Transactions on Image Processing Vol. 2*, No. 1 (Jan. 1993), pp. 27–40.