Universidad Miguel Hernández de Elche

Departamento de Ingeniería de Sistemas y Automática

Transmisión Robusta y Eficiente de Vídeo en Redes Vehiculares

Tesis Doctoral

Memoria presentada para optar al grado de Doctor por:
D. Pablo José Piñol Peral

Dirigida por:
D. Manuel José Pérez Malumbres
D. Otoniel Mario López Granado

UNIVERSITAS
Miguel
Hernández

D. MANUEL JOSÉ PÉREZ MALUMBRES, Catedrático de Universidad de la Universidad Miguel Hernández de Elche y D. OTONIEL MARIO LÓPEZ GRANADO, Profesor Contratado Doctor de la Universidad Miguel Hernández de Elche,

CERTIFICAN:

Que la presente memoria *Transmisión Robusta y Eficiente de Vídeo en Redes Vehiculares*, ha sido realizada bajo su dirección, en el Departamento de Ingeniería de Sistemas y Automática de la Universidad Miguel Hernández de Elche, por el Licenciado D. PABLO JOSÉ PIÑOL PERAL, y constituye su tesis para optar al grado de Doctor.

Para que conste, en cumplimiento de la legislación vigente, autorizan la presentación de la referida tesis doctoral ante la Comisión de Doctorado de la Universidad Miguel Hernández de Elche, firmando el presente certificado.

Elche, 27 de Mayo de 2015

Fdo. D. Manuel J. Pérez Malumbres Fdo. D. Otoniel M. López Granado

UNIVERSITAS
Miguel
Hernández

D. LUIS MIGUEL JIMÉNEZ GARCÍA, Profesor Titular de Universidad y director del Departamento de Ingeniería de Sistemas y Automática de la Universidad Miguel Hernández de Elche,

CERTIFICA:

Que la presente memoria *Transmisión Robusta y Eficiente de Vídeo en Redes Vehiculares*, realizada bajo la dirección de D. MANUEL JOSÉ PÉREZ MALUMBRES y D. OTONIEL MARIO LÓPEZ GRANADO, en el Departamento de Ingeniería de Sistemas y Automática de la Universidad Miguel Hernández de Elche, por el Licenciado D. PABLO JOSÉ PIÑOL PERAL, constituye su tesis para optar al grado de Doctor.

Para que conste, en cumplimiento de la legislación vigente, autoriza la presentación de la referida tesis doctoral ante la Comisión de Doctorado de la Universidad Miguel Hernández de Elche, firmando el presente certificado.

Elche, 27 de Mayo de 2015

Fdo. D. Luis Miguel Jiménez García

UNIVERSITAS
Miguel
Hernández

Agradecimientos

"Lo importante es quererse..."
Isabel Cuevas Rubio

Dedicado a mi familia. Os quiero y lo sabéis.

Quiero agradecer a mis directores de tesis su labor de dirección de este trabajo, así como su amistad.

Quiero agradecer a mis compañeros (y antiguos compañeros) de la universidad Miguel Hernández de Elche su gran colaboración en este trabajo y su buena compañía durante todos estos años, así como su amistad.

Quiero agradecer a mis compañeros de la Universidad Politécnica de Valencia y de la Universidad de Zaragoza su ayuda en distintas etapas de la realización de esta tesis, así como su amistad.

Quiero agradecer a mi familia todo el apoyo que me han brindado y el esfuerzo que han realizado conmigo para que pudiera desarrollar el trabajo que hoy culmina.

Gracias.

Resumen

En el futuro, las redes vehiculares serán tan comunes como actualmente lo son los *smartphones*. Los vehículos estarán equipados con multitud de sensores de todo tipo, con una cierta capacidad de proceso y la posibilidad de comunicarse entre sí y con la infraestructura disponible, siendo capaces de recoger información, elaborarla y distribuirla. Una de las tecnologías que serán críticas para la implantación de servicios de información en las redes vehiculares es la transmisión de vídeo. La transmisión de vídeo está asociada a aplicaciones que van desde el entretenimiento digital hasta las aplicaciones de seguridad, como la vídeo-llamada de emergencia, pasando por otras como la publicidad contextual, o la información turística.

Aunque la transmisión de vídeo en redes vehiculares puede tener multitud de usos beneficiosos, se encuentra con dos grandes dificultades. Por una parte, debido a su naturaleza inalámbrica y a la movilidad de sus nodos, las redes vehiculares sufren particularmente un mayor impacto en la comunicación debido a la pérdida de paquetes. Por otra parte, el vídeo digital tiene unos altos requerimientos en cuanto a recursos, tanto por el alto volumen de datos que maneja como por los grandes requerimientos de cómputo necesarios para poder comprimir el vídeo a transmitir. Esta tesis aborda estos dos problemas, teniendo como objetivo principal la viabilidad de la transmisión de vídeo en estas redes. Por una parte busca la eficiencia en el procesamiento, proponiendo técnicas de computación paralela para acelerar la compresión de vídeo y por otra parte busca la robustez frente a la pérdida de datos, proponiendo soluciones para mejorar la calidad de experiencia del usuario.

En cuanto a la utilización de las técnicas de paralelismo para acelerar la codificación de vídeo, se han diseñado dos aproximaciones distintas. En la primera aproximación se aplica una paralelización de alto nivel, en la que las estrategias de paralelismo están diseñadas a nivel de grupo de cuadros o imágenes. En esta aproximación se proponen diferentes agrupaciones de los cuadros que componen una secuencia y cada grupo se asigna a un procesador distinto para que sean procesados en paralelo. La segunda aproximación es a bajo nivel y está basada en dividir cada cuadro de la secuencia en varias regiones y asignar cada región a un procesador para que trabajen en paralelo.

Para cada una de las diferentes aproximaciones se ha evaluado la capacidad de aceleración en la codificación del vídeo así como la distorsión introducida.

Para añadir robustez a la transmisión de vídeo (y en línea con la segunda de las dos aproximaciones de computación paralela) se ha dividido cada cuadro en varias regiones, con el objetivo de ajustar el tamaño de cada área codificada al tamaño máximo de transmisión de datos, lo cual mejora la robustez en cuanto a la pérdida de paquetes, pero implica aumentar la tasa de paquetes por segundo a transmitir, lo que puede implicar una mayor tasa de pérdidas. Mejorar la robustez global requiere buscar un equilibrio para que la solución no se convierta en el problema. También se han propuesto técnicas de corrección de errores a nivel de red para paliar la incidencia de las pérdidas en la decodificación de los datos. Además, se han tenido en cuenta las características específicas de la codificación y decodificación del vídeo, tales como los modos de compresión, con el fin de mejorar la calidad final del vídeo reconstruído.

Esta tesis se ha realizado bajo la modalidad de presentación de tesis doctorales con un conjunto de publicaciones, recogida en la normativa de la Universidad Miguel Hernández de Elche. En cumplimiento de dicha normativa se han incorporado las publicaciones que la componen como anexo y se han incluído las secciones correspondientes a la descripción general de los trabajos, el resumen global de los resultados obtenidos y las conclusiones finales.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción general

Índice

1.1. Motivación

La línea de investigación en la que se encuadra la presente tesis es la transmisión de datos multimedia en redes inalámbricas. Se sitúa, más específicamente, dentro del campo de la transmisión de vídeo en redes vehiculares.

Las redes vehiculares [1] se componen de nodos móviles (vehículos) que circulan por calles y carreteras, así como de infraestructura fija que permite proporcionar conectividad con otras redes. Los vehículos están equipados con unidades de a bordo, *On Board Units* (OBU), que son las encargadas de procesar los datos que se recogen en los distintos sensores instalados en el vehículo así como de realizar las tareas de comunicación. Por su parte, la infraestructura fija se compone de unidades colocadas al lado de carreteras y calles, *Road Side Units* (RSU), así como de las infraestructuras de comunicaciones de otros tipos de redes como pueden ser las de telefonía móvil (3G, 4G, ...), redes satelitales, redes WiMAX (*Worldwide Interoperability for Microwave Access*) y otras.

Debido a que la conectividad entre los nodos que forman una red vehicular es generalmente esporádica (breve y en continuo cambio), a estas redes se les denomina redes vehiculares *ad hoc*, *Vehicular Ad Hoc Networks* (VANET) [2] y son consideradas como un tipo específico dentro de la categoría de redes móviles *ad hoc*, *Mobile Ad Hoc Networks* (MANET).

Las VANETs son las redes que proporcionan el soporte de comunicaciones a los denominados sistemas inteligentes de transporte, *Intelligent Transportation Systems* (ITS) [3][4][5]. El objetivo de los ITS es aprovechar la capacidad de recogida de datos, de procesamiento de información y de comunicaciones de los vehículos para proporcionar servicios de alto nivel a distintos destinatarios (pasajeros del transporte público o privado, compañías que dispongan de flotas de vehículos, gobiernos locales o nacionales, etc.). El rango del tipo de servicios proporcionados por los ITS es muy amplio, cumpliendo propósitos tan diversos como la seguridad de las personas, la optimización de rutas, la reducción en el consumo de combustible y en las emisiones de CO_2, el entretenimiento, etc. Hay multitud de aplicaciones de los ITS que pueden beneficiarse de la transmisión de vídeo en las VANETs, como la información turística, la publicidad contextual, el vídeo bajo demanda, la video-conferencia, la monitorización de condiciones de tráfico o metereológicas, la video-llamada de emergencia, etc. A pesar de que puede resultar muy útil la transmisión de vídeo en las redes vehiculares, llevar a cabo esta tarea de forma satisfactoria resulta altamente complicado debido por una parte a las características del vídeo digital y por otra parte a las

características de dichas redes.

El vídeo digital es un tipo de datos que lleva asociados unos requerimientos muy altos. Por una parte la cantidad de datos que se generan al capturar vídeo digital es inmensa. Esto hace imprescindible un proceso de compresión de dichos datos [6][7] para poder trabajar con ellos (almacenarlos, transmitirlos,...). Incluso usando los métodos de compresión y debido a las continuamente crecientes resoluciones y tasas de imágenes por segundo utilizadas, la cantidad de datos a manejar, respecto a otro tipo de aplicaciones, es grande. Para tratar de reducir al máximo el tamaño de los datos de una secuencia de vídeo manteniendo un cierto nivel de calidad se hace necesario utilizar los codificadores de vídeo más recientes, que permiten conseguir las mejores tasas de compresión. El uso de estos codificadores consigue aumentar la eficiencia en la compresión de los datos, con la contrapartida asociada de un incremento en los requisitos de procesamiento debido a la gran complejidad computacional de los mismos.

Por su parte, las redes vehiculares son entornos hostiles, tanto por la movilidad de sus nodos (topología de red en continuo cambio, aislamiento en redes con nodos dispersos, etc.), como por su naturaleza inalámbrica (obstáculos que atenúan la señal, congestión en redes muy densas, ancho de banda limitado, etc.). Una de las consecuencias directas de este contexto es la pérdida de paquetes de datos.

En general, los datos comprimidos son muy sensibles a las pérdidas, ya que suelen tener grandes interdependencias entre sí y eso conlleva a que la pérdida de una parte de los datos implique que la información recibida correctamente no sea de ninguna utilidad o bien se recupere de forma errónea, introduciendo un cierto nivel de degradación en la información. El vídeo codificado es especialmente sensible a las pérdidas de paquetes por varios motivos. Entre ellos podemos citar los siguientes: (1) La estructura sintáctica de los datos comprimidos. Los datos de una secuencia de vídeo comprimida están estructurados internamente en unidades sintácticas. Si una unidad sintáctica está fragmentada en varios paquetes de datos, la pérdida de uno de estos paquetes impedirá la utilización de dicha unidad sintáctica en la decodificación del vídeo. Por lo tanto, la pérdida real de un único paquete implica la pérdida efectiva de todos los paquetes que componen dicha unidad sintáctica. (2) La codificación entrópica usada. Debido al uso de la codificación entrópica, ante la pérdida de parte de los datos o de errores en los mismos, puede que se produzca una decodificación errónea de la información que sí ha llegado correctamente. (3) La explotación de la redundancia espacial y/o temporal de las secuencias de vídeo para la compresión de los datos. Para aprovechar la redundancia espacial encontrada en un fotograma, se codifican

algunas regiones de la imagen de forma incremental (sólo se codifica el resíduo), usando predicciones basadas en las regiones previamente codificadas de la misma imagen. La forma de aprovechar la redundancia temporal de las secuencias es codificarlas de forma incremental, basándose en regiones similares que se encuentran en otras imágenes (anteriores o posteriores a la imagen actual). Esta forma de compresión conlleva que la pérdida de un paquete de datos (y por consiguiente de una región de una imagen) afectará desfavorablemente a la decodificación tanto de dicha región como de las regiones que hayan utilizado ese fragmento de imagen como referencia. Y no sólo eso, sino que las regiones que han sido decodificadas erróneamente extenderán los errores a las regiones que han utilizado a éstas como referencia, y así sucesivamente. Por tanto, la pérdida de un único paquete de datos puede crear un efecto avalancha (o efecto dominó) que afecte a multitud de regiones de la misma imagen y a multitud de otras imágenes que componen el vídeo.

Ante este escenario surgen dos necesidades para poder llevar a cabo adecuadamente la tarea de la transmisión de vídeo en redes vehiculares: acelerar los mecanismos de compresión de vídeo para aliviar el incremento en la complejidad computacional de los codificadores actuales y añadir mecanismos de protección a las secuencias de vídeo codificado para que sean resistentes ante las pérdidas de paquetes y así minimizar la pérdida de calidad final en el vídeo reconstruído.

1.2. Objetivos de la tesis

El objetivo general de la tesis es el siguiente:

- Proponer métodos eficientes y robustos para la transmisión de vídeo en redes vehiculares.

Este objetivo general se desglosa en los siguientes objetivos específicos:

- Diseñar y evaluar técnicas de computación paralela específicas para la codificación de vídeo, que permitan la aceleración de dicho proceso y así paliar el alto coste computacional de los codificadores actuales.

- Diseñar y evaluar mecanismos que garanticen la transmisión robusta del vídeo en las redes vehiculares, basados en las características de los codificadores de vídeo más recientes.

- Proponer y evaluar mecanismos de protección ante errores a nivel de red para minimizar los efectos de las pérdidas de paquetes que se producen en las redes vehiculares.

1.3. Publicaciones

La presente tesis ha sido realizada mediante la modalidad de presentación de tesis doctorales con un conjunto de publicaciones, recogida en la normativa de la Universidad Miguel Hernández de Elche. En dicha normativa se indica que la tesis debe incorporar una introducción general donde se presenten los trabajos y un resumen global de los resultados obtenidos y de las conclusiones finales, así como un anexo con los trabajos publicados en su idioma original. En cumplimiento de dicha normativa, las publicaciones que conforman esta tesis se han incorporado en el Anexo II de la presente memoria. A continuación se proporciona una descripción general de dichos trabajos y en los próximos capítulos se presentarán los resultados obtenidos así como las conclusiones finales.

Las publicaciones detalladas a continuación proponen métodos que utilizan el codificador *High Efficiency Video Coding* (HEVC) [8] para la transmisión de vídeo en redes vehiculares de una forma eficiente y robusta. HEVC es el estándar de codificación de vídeo más reciente propuesto por el equipo JCT-VC (*Joint Collaborative Team on Video Coding*) en Enero de 2013. El objetivo del JCT-VC era el desarrollo de un nuevo estándar de codificación de vídeo que consiguiera duplicar la eficiencia del anterior estándar de codificación, H.264/AVC (*Advanced Video Coding*) [9]. Esto significa una reducción del 50 % en la tasa de *bits* (*bit rate*) para un mismo nivel de calidad. El codificador HEVC consigue eficiencias muy cercanas a las pretendidas, y esto es fundamental para poder trabajar con las secuencias de vídeo actuales que utilizan resoluciones y tasas de cuadros por segundo cada vez más grandes. Como se ha expuesto anteriormente, el aumento tan sorprendente en la eficiencia de la codificación del nuevo estándar viene acompañado de un incremento en la complejidad computacional de varios órdenes de magnitud.

1.3.1. *Parallel strategies analysis over the HEVC encoder*

En este trabajo se proponen varias estrategias de paralelización para llevar a cabo la compresión de vídeo usando el codificador HEVC. En este caso se establecen los mecanismos de paralelización a nivel de GOP (*Group Of*

Pictures). Un GOP es un conjunto de imágenes consecutivas dentro de una secuencia de vídeo que incluye al menos un cuadro de tipo I (*intra*). Los cuadros de tipo I se codifican sin usar como referencia otros cuadros, por lo tanto pueden ser decodificados independientemente del resto de las imágenes de la secuencia de vídeo. Dentro de un cuadro I se puede usar la redundancia espacial para codificar partes de la imagen de forma incremental (predicciones y resíduo). Los GOPs, además de contar con uno o más cuadros de tipo I, pueden contener cuadros de tipo P y/o de tipo B. Los cuadros de tipo P (*predictive*) usan otro cuadro dentro de la secuencia para realizar la estimación y compensación de movimiento y así explotar la redundancia temporal existente. Para ello buscan, en el cuadro de referencia, regiones similares a la región que se está codificando en el cuadro actual para codificarla de forma incremental (vectores de movimiento y resíduo). Los cuadros de tipo B (*bi-directional predictive*) utilizan un método muy similar a los cuadros P, pero pueden usar como predicción la interpolación de dos regiones ubicadas en dos cuadros de referencia distintos. En el codificador HEVC los cuadros de referencia para los cuadros de tipo P y B pueden encontrarse antes o después del cuadro actual.

Para la evaluación de las estrategias se han usado los modos de compresión *All Intra* (AI) y *Low-delay B* (LB). En el modo AI se codifican todas las imágenes de una secuencia como cuadros I. En el modo LB se codifica el primer cuadro de tipo I y el resto de tipo B. El calificativo *low-delay* (bajo retardo) se utiliza para indicar que el par de cuadros de referencia que usa cada imagen siempre se toman de los anteriores (en orden de reproducción). Esto implica que el orden de codificación (y también de decodificación) de los cuadros coincide con el orden de reproducción de dichos cuadros. Si se usaran cuadros de referencia futuros se introduciría un cierto retardo ya que al codificar (ó decodificar) tendríamos que esperar a que esos cuadros futuros estuvieran codificados (ó decodificados) para poder usarlos como referencia. En el caso del modo LB, a la hora de codificar (o decodificar) un cuadro ya se disponen de todos los cuadros de referencia y por lo tanto el retardo es mínimo.

Se proponen cuatro estrategias de paralelización. Tres de ellas se aplican al modo LB, tomando un tamaño de GOP de 4 cuadros. La cuarta aproximación se aplica al modo AI, donde el GOP tiene un tamaño de 1. Se han obtenido resultados usando una plataforma *multicore* con un esquema de memoria compartida, utilizando varias secuencias de vídeo (con distintas resoluciones y tasas de cuadros por segundo) y variando el número de procesos que trabajan en paralelo. Por una parte, se ha evaluado la eficiencia computacional de las estrategias paralelas midiendo los tiempos de compresión y la aceleración en términos relativos (*speed-up*). Por otra parte, se ha comparado cómo se

comportan cada una de las estrategias en cuanto a la tasa de compresión y a la calidad del vídeo reconstruído. La explicación detallada de las propuestas así como la evaluación de las mismas se recogen en el apartado 2.2.1.

1.3.2. *Slice-based parallel approach for HEVC encoder*

En este trabajo se refina la paralelización del algoritmo de compresión para establecerlo a nivel de *slice*. Una *slice* es un fragmento codificado de una imagen, que puede ser decodificado de forma independiente (con respecto al resto de *slices* de la misma imagen). Esto impide que se puedan usar las regiones de la misma imagen pertenecientes a otras *slices* como predicciones para aprovechar la redundancia espacial. Tampoco se puede usar la información de otras *slices* del mismo cuadro como predicción de otros valores, como los vectores de movimiento. Esta imposición, que no permite explotar la correlación existente entre regiones de una misma imagen (cuando pertenecen a *slices* distintas) tiene una cierta penalización sobre la eficiencia de la codificación (en cuanto a la tasa de compresión). Este elemento sintáctico se introdujo en los codificadores de vídeo con el objeto de proporcionar resistencia ante errores, ya que al tratarse de regiones independientes, la pérdida de una *slice* no impide la correcta decodificación del resto de *slices* del mismo cuadro, si se reciben correctamente. A pesar de no haber sido diseñadas con el objetivo de facilitar la computación paralela, las *slices*, de forma intrínseca, permiten la aplicación de estrategias de paralelización debido a su independecia. Pero el nivel de paralelismo que proporcionan está limitado. En los cuadros de tipo P y B, a la hora de codificar una *slice*, se necesita que el cuadro (o cuadros) de referencia que utilizará dicha *slice* estén disponibles en su totalidad. Esto implica que el número máximo de procesos en paralelo a utilizar viene marcado por el número de *slices* de cada cuadro y que deberá existir obligatoriamente un mecanismo de sincronización de procesos (al menos cuando se usan cuadros de tipo P ó B).

Para la evaluación del esquema de paralelización propuesto se han usado los dos modos de compresión citados anteriormente (AI y LB), además de los modos *Low-delay P* (LP) y *Random Access* (RA). El modo LP es similar al modo LB, pero en lugar de utilizar cuadros de tipo B se usan cuadros de tipo P. El modo RA consiste en insertar un cuadro I aproximadamente cada segundo de vídeo y el resto de cuadros de tipo B, los cuales usan como referencias otros cuadros anteriores y posteriores al actual. En este modo de compresión se introduce un cierto retardo, tanto en la codificación como en la decodificación, debido a que el orden de codificación no coincide con el orden de reproducción y hay que esperar a que los cuadros de referencia (futuros)

hayan sido codificados.

Se han obtenido resultados evaluando distintas secuencias de vídeo con distintos números de *slices* por cuadro. En esta propuesta el número de procesos va directamente relacionado con el número de *slices*. Se han medido la eficiencia tanto a nivel de computación paralela como a nivel de flujo de vídeo comprimido (calidad y tasa de compresión). Los resultados más significativos se recogen en el apartado 2.2.2.

1.3.3. *Protection of HEVC Video Delivery in Vehicular Networks with RaptorQ Codes*

En este trabajo se proponen y evalúan técnicas para transmitir vídeo de forma robusta en las redes vehiculares. Como se ha dicho anteriormente las *slices* fueron introducidas en la codificación de vídeo para proporcionar resistencia ante errores, aprovechando su propiedad principal: ser decodificables de forma independiente al resto de *slices* del mismo cuadro. Esta característica, como se ha citado anteriormente, permite aislar (en cierta medida) las pérdidas de paquetes. Además, mediante el uso de *slices* se puede adecuar el tamaño de las unidades sintácticas del vídeo codificado al tamaño máximo de datos de red, *Maximum Transfer Unit* (MTU), aliviando la vulnerabilidad que aparece cuando una unidad sintáctica está muy fragmentada. En el artículo se estudia, en primer lugar, cómo afecta el número de *slices* por cuadro a la calidad (y a la tasa de *bits*) del vídeo codificado. Se observa el impacto que produce el uso de *slices* en el vídeo, en un entorno ideal, donde no se producen pérdidas de paquetes. También se estudian otros indicadores que varían con el número de *slices* por cuadro, como son la tasa media de paquetes de red por cada *slice* y el número de paquetes por segundo a transmitir. Estas dos características influyen en la pérdida de la información útil en la transmisión. Si la proporción de paquetes de red por cada *slice* es demasiado alta entonces un porcentaje bajo de pérdidas de paquetes puede producir un porcentaje alto de pérdidas de *slices*. El número de paquetes por segundo a transmitir repercute directamente en la saturación del canal de transmisión de datos, con lo que puede aumentar la tasa de paquetes perdidos.

Se han utilizado varias herramientas *software* de terceros, así como desarrollos propios, para simular una red vehicular en un escenario realista (usando mapas reales) y poder evaluar el comportamiento del vídeo codificado usando distintos números de *slices* por cuadro. Además se ha introducido un mecanismo de protección de paquetes a nivel de red (Forward Error Correction - FEC) que mediante el uso de paquetes redundantes permite la recuperación de un porcentaje de los paquetes perdidos.

En el apartado 2.2.3 se muestran los resultados más relevantes de esta publicación.

Los mecanismos de protección propuestos en este artículo permiten aprovechar directamente las ventajas de la propuesta de paralelización a nivel de *slice*. Además, la propuesta de paralelización a nivel de GOP siempre es aplicable por encontrarse en un nivel superior. En el apartado de trabajo futuro se marcan las líneas maestras para la ampliación de la investigación realizada en estas tres publicaciones.

Capítulo 2

Resultados de investigación

Índice

2.1. Materiales y métodos

Para la obtención de los resultados, en la investigación se han usado los siguientes materiales y métodos.

- ***Software* de referencia del codificador HEVC**
  Para la codificación de las secuencias de vídeo se ha utilizado el *software* de referencia del codificador HEVC [10]. En dicho *software* se han realizado las modificaciones correspondientes para implementar las propuestas de paralelización. También se han realizado las modificaciones para que pueda trabajar en formato RTP (*Real-time Transport Protocol*) y para que ante una secuencia de vídeo comprimida en la que faltan los paquetes perdidos el decodificador no se interrumpa. En el *software* de referencia se encuentran los ficheros de configuración de los cuatro modos de compresión citados: AI, LP, LB, RA.

- **Secuencias de vídeo**
  Se han utilizado las siguientes secuencias de vídeo que pertenecen al conjunto de secuencias especificadas en las condiciones comunes de evaluación de HEVC [11].

Secuencia	Resolución	Cuadros por segundo
Traffic	2560x1600	30
BQ Terrace	1920x1080	60
Four People	1280x720	60
Race Horses	832x480	30
Basketball Pass	416x240	50

Tabla 2.1: Secuencias de vídeo usadas en la investigación.

- **Plataforma *multicore***
  Plataforma con dos *hexacores* Intel XEON X5660, a 2.8 GHz, con 12 MB de caché por procesador y 48 GB de RAM. Sistema operativo CentOS Linux 5.6 para arquitecturas x86 de 64 *bits*.

- **Interfaz de programación de aplicaciones OpenMP**
  La paralelización del codificador HEVC se ha llevado a cabo mediante la utilización del interfaz de programación de aplicaciones OpenMP (*Open Multi-Processing*) [12] y el compilador *g*++ v.4.1.2.

- **Repositorio de mapas geográficos OpenStreetMap**
  OpenStreetMap [13] es un mapa del mundo de uso libre bajo una licencia

abierta (similar a Wikipedia) creado por voluntarios de todo el mundo. Se ha utilizado OpenStreetMap para obtener las definiciones de escenarios reales en los que lanzar las simulaciones de las redes vehiculares.

- **Simulador de movilidad de vehículos SUMO**
  Para la simulación del movimiento de los vehículos en los escenarios se ha utilizado el simulador SUMO (*Simulation of Urban MObility*) [14]. SUMO permite la comunicación con otros simuladores (como por ejemplo un simulador de redes) a través del interfaz TraCI (*TRAffic Control Interface*) [15]. Para convertir los datos geográficos descargados de OpenStreetMap al formato que maneja SUMO se han usado las herramientas *netconvert* y *polyconvert* (que forman parte del simulador).

- **Simulación de redes vehiculares: OMNeT++, MiXim, Veins**
  OMNeT++ [16] es un entorno para el desarrollo de simuladores. MiXiM [17] es un simulador de redes inalámbricas (tanto móviles como estáticas) desarrollado bajo el entorno OMNeT++. Veins [18] es otro simulador desarrollado bajo OMNeT++ que trabaja en colaboración con MiXiM aportando los protocolos creados específicamente para las redes vehiculares, como son el protocolo IEEE 802.11p [19] y la familia de estándares IEEE 1609 [20]. Además de estos tres programas, que permiten la simulación de una red vehicular, se ha realizado un desarrollo propio que permite la inyección de trazas de vídeo y la grabación de ficheros con los paquetes recibidos, así como la recolección de diversas estadísticas, como el porcentaje de paquetes perdidos y la distribución de los mismos.

- **Herramienta de protección de paquetes**
  Para la protección de los paquetes a nivel de red hemos utilizado una herramienta FEC (*Forward Error Correction*), seleccionando el *software Qualcomm (R) RaptorQ Evaluation Kit* [21]. Quiero mostrar mi agradecimiento a Qualcomm por habernos facilitado dicho *software* para las investigaciones.

2.2. Resultados

Resumen global de los resultados de investigación y la discusión de los mismos.

2.2.1. Paralelización a nivel de GOP

La paralelización a nivel de GOP consiste en dividir la secuencia de vídeo en grupos de cuadros, de forma que el procesamiento de cada GOP se realice de forma completamente independiente del resto. En general, esta aproximación proporcionará un buen rendimiento en cuanto a aceleración de la codificación del vídeo, pero dependiendo de cómo seleccionemos la estructura de GOPs y cómo hagamos la distribución de la codificación a cada proceso, la eficiencia en la compresión se puede ver afectada, principalmente por romper las dependencias entre cuadros contiguos.

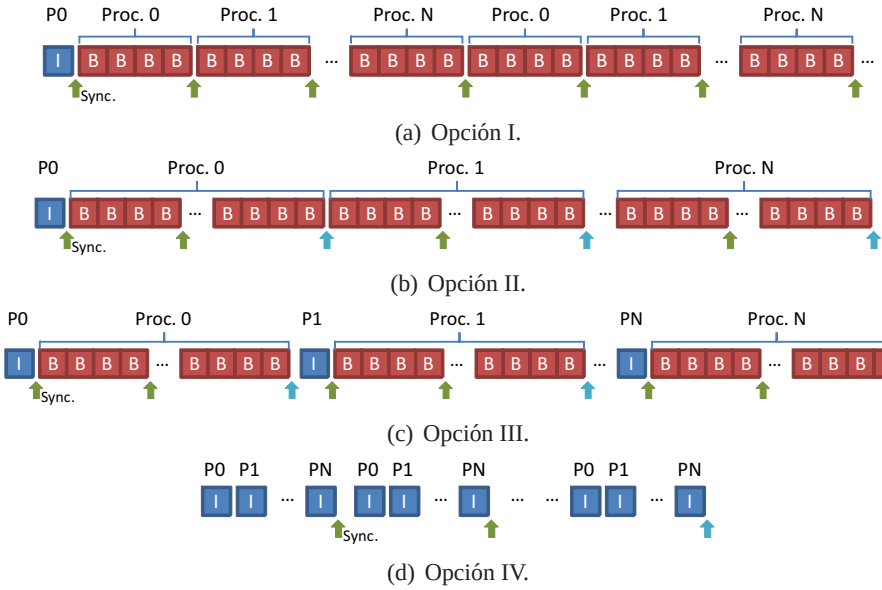
A nivel de GOP se han propuesto las siguientes cuatro estrategias de paralelización (en la Figura 2.1 están representadas gráficamente).

- **Opción I** (*modo LB*): En esta opción se asigna secuencialmente cada GOP a un proceso en paralelo, así, cada proceso codificará un GOP aislado.
- **Opción II** (*modo LB*): En esta opción se divide la secuencia en tantas partes como número de procesos en paralelo, de esta forma cada proceso codificará un bloque de GOPs contiguos.
- **Opción III** (*modo LB*): Esta opción es similar a la opción II, salvo que cada proceso comienza la codificación de su bloque insertando un cuadro de tipo I.
- **Opción IV** (*modo AI*): Esta opción es similar a la opción I donde cada GOP se asigna secuencialmente a un proceso, pero en este caso el GOP consiste en un único cuadro de tipo I.

En las figuras 2.2, 2.3, 2.4 y 2.5 se presentan los tiempos de computación para las opciones I, II, III y IV, respectivamente, codificando 240 cuadros de las secuencias *Basketball Pass*, *Race Horses* y *Four People*. Los resultados muestran un buen comportamiento paralelo en todos los casos. Notar que los valores para 1 proceso son los correspondientes al algoritmo secuencial.

En las Figuras 2.6, 2.7, 2.8 y 2.9 se muestra la aceleración (*speed-up*) obtenida para los mismos casos. Se observa que se obtienen valores casi ideales en algunos casos y que la aceleración se incrementa a la vez que aumentamos el número de procesos. El ratio de incremento se mantiene prácticamente constante para todas las combinaciones de número de procesos y de secuencias.

En la Figura 2.10 se compara la aceleración obtenida por los algoritmos paralelos propuestos al codificar 240 cuadros de la secuencia *BQ Terrace*. La



(a) Opción I.

(b) Opción II.

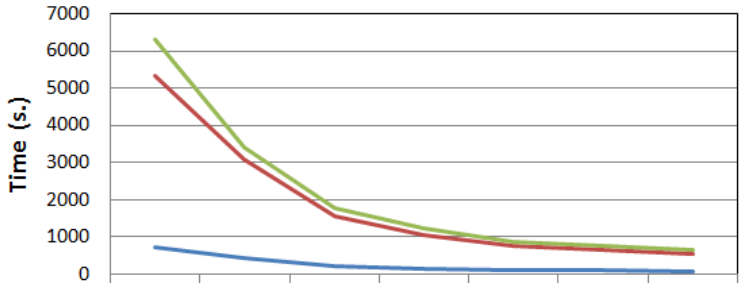
(c) Opción III.

(d) Opción IV.

Figura 2.1: Propuestas de paralelización a nivel de GOP.

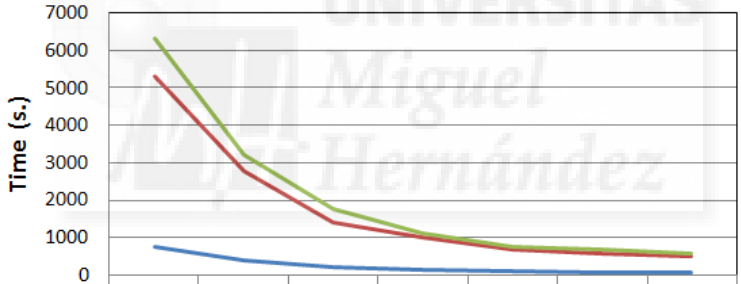
Opción III obtiene el mejor rendimiento de las que usan el modo de codificación LB (este comportamiento es similar para el resto de secuencias). Cabe destacar que la Opción IV (modo AI) obtiene una aceleración similar a la Opción III (modo LB).

La Figura 2.11 muestra la evolución de la tasa de *bits* y de la distorsión, usando la métrica PSNR (*Peak Signal-to-Noise Ratio*), como función del número de núcleos de procesamiento usados, para la secuencia *Race Horses* y las Opciones I, II y III (modo LB). Las tres opciones producen un incremento en la tasa de *bits* cuando el número de procesos aumenta. Se puede observar claramente que el incremento en la tasa de *bits* y el decremento en calidad que se produce en la Opción I hacen que esta opción sea directamente descartable, ya que aunque su comportamiento en cuanto a la eficiencia computacional es el adecuado, la eficiencia en la compresión del vídeo se ve penalizada gravemente. Este efecto se debe a que la Opción I cambia drásticamente el orden de codificación de los cuadros, e impide aprovechar de forma óptima la redundancia temporal del vídeo (que es mucho mayor en los cuadros contiguos). Si comparamos las Opciones II y III se observa que en ambos casos se produce un aumento relativamente controlado de la tasa de *bits* cuando el número de procesos aumenta, pero que en la Opción III este incremento produce una mejora en PSNR mientras que en la Opción II se produce un decremento en la calidad. Por este motivo, la Opción III muestra



	1P	2P	4P	6P	8P	10P	12P
BP	744.29	431.63	232.14	165.49	122.94	111.42	79.61
RH	5309.82	3062.06	1566.11	1073.81	764.20	656.95	560.21
FP	6314.44	3396.41	1770.66	1217.59	864.83	772.29	650.17

Figura 2.2: Tiempo de computación obtenido usando la **Opción I** para 240 cuadros de las secuencias *Basketball Pass* (BP), *Race Horses* (RH) y *Four People* (FP).

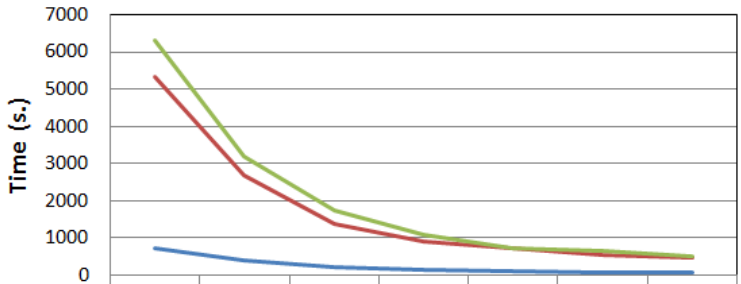


	1P	2P	4P	6P	8P	10P	12P
BP	744.29	401.12	208.80	138.41	111.25	83.80	77.50
RH	5309.82	2781.76	1403.59	985.08	663.24	574.57	488.77
FP	6314.44	3228.47	1770.66	1121.99	778.21	687.32	580.27

Figura 2.3: Tiempo de computación obtenido usando la **Opción II** para 240 cuadros de las secuencias *Basketball Pass* (BP), *Race Horses* (RH) y *Four People* (FP).

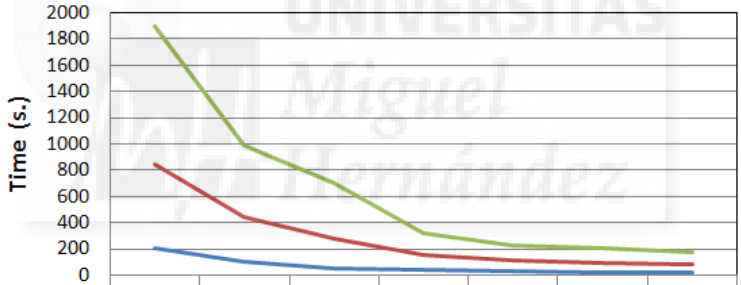
un mejor comportamiento en cuanto a la eficiencia en la codificación del vídeo.

Por último, se han modificado los ficheros de configuración para el modo LB de las versiones secuencial y paralela aplicadas a la Opción III para que produzcan los mismos valores de PSNR y de tasa de *bits*. En la Figura 2.12, la



	1P	2P	4P	6P	8P	10P	12P
BP	744.29	395.68	203.69	138.24	112.81	80.01	69.48
RH	5309.82	2674.27	1379.97	909.00	732.74	546.16	464.90
FP	6314.44	3204.06	1747.19	1089.31	744.01	656.15	524.92

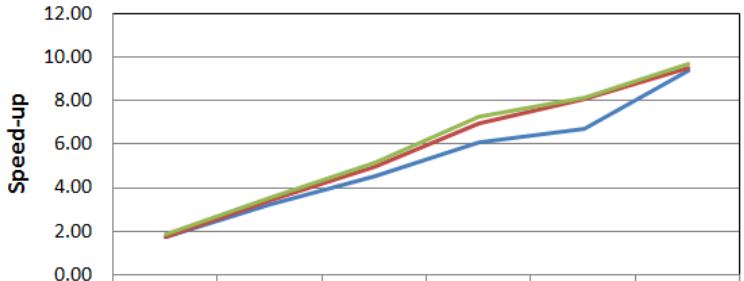
Figura 2.4: Tiempo de computación obtenido usando la **Opción III** para 240 cuadros de las secuencias *Basketball Pass* (BP), *Race Horses* (RH) y *Four People* (FP).



	1P	2P	4P	6P	8P	10P	12P
BP	201.64	101.63	51.98	35.68	27.43	22.06	19.41
RH	841.10	438.96	275.91	151.69	113.39	94.01	83.09
FP	1899.84	988.37	701.78	321.93	222.08	205.35	173.25

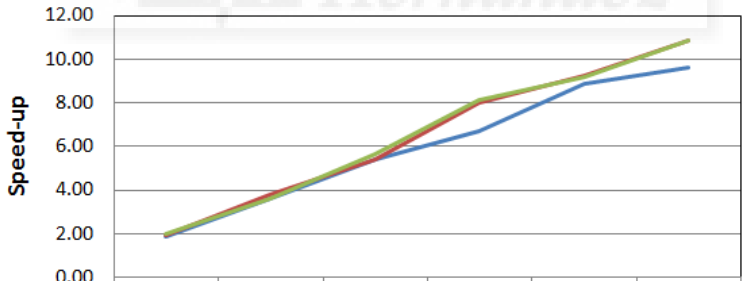
Figura 2.5: Tiempo de computación obtenido usando la **Opción IV** para 240 cuadros de las secuencias *Basketball Pass* (BP), *Race Horses* (RH) y *Four People* (FP).

curva EQUIV muestra la aceleración cuando se obtienen vídeos comprimidos equivalentes. La curva NON_EQUIV muestra la aceleración cuando se obtienen vídeos comprimidos ligeramente distintos (sin realizar el ajuste en los ficheros de configuración). Se puede observar que en el modo EQUIV la Opción III sigue obteniendo aceleraciones con una buena eficiencia computacional, aunque sean ligeramente inferiores al modo NON_EQUIV.



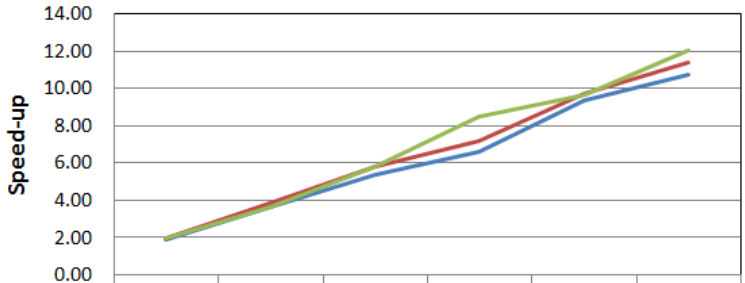
	2P	4P	6P	8P	10P	12P
BP	1.72	3.21	4.50	6.05	6.68	9.35
RH	1.73	3.39	4.94	6.95	8.08	9.48
FP	1.86	3.57	5.19	7.30	8.18	9.71

Figura 2.6: Aceleración (*speed-up*) obtenida usando la **Opción I** para 240 cuadros de las secuencias *Basketball Pass* (BP), *Race Horses* (RH) y *Four People* (FP).



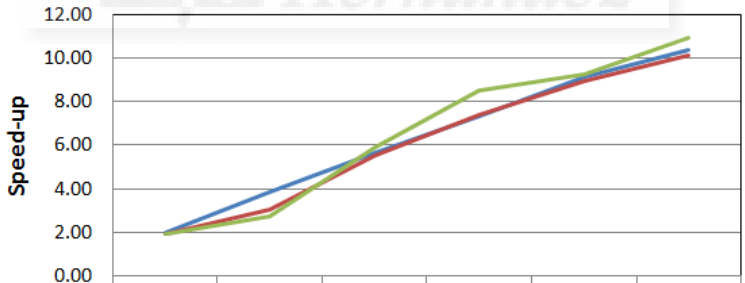
	2P	4P	6P	8P	10P	12P
BP	1.86	3.56	5.38	6.69	8.88	9.60
RH	1.91	3.78	5.39	8.01	9.24	10.86
FP	1.96	3.57	5.63	8.11	9.19	10.88

Figura 2.7: Aceleración (*speed-up*) obtenida usando la **Opción II** para 240 cuadros de las secuencias *Basketball Pass* (BP), *Race Horses* (RH) y *Four People* (FP).



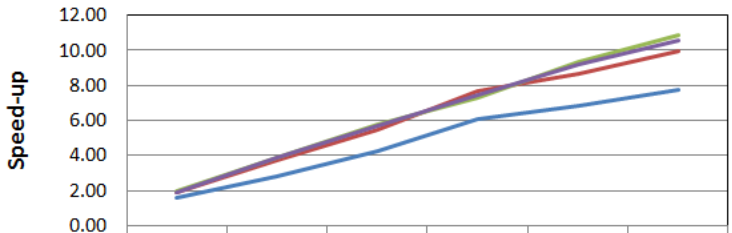
	2P	4P	6P	8P	10P	12P
BP	1.88	3.65	5.38	6.60	9.30	10.71
RH	1.99	3.85	5.84	7.25	9.72	11.42
FP	1.97	3.61	5.80	8.49	9.62	12.03

Figura 2.8: Aceleración (*speed-up*) obtenida usando la **Opción III** para 240 cuadros de las secuencias *Basketball Pass* (BP), *Race Horses* (RH) y *Four People* (FP).



	2P	4P	6P	8P	10P	12P
BP	1.98	3.88	5.65	7.35	9.14	10.39
RH	1.92	3.05	5.54	7.42	8.95	10.12
FP	1.92	2.71	5.90	8.55	9.25	10.97

Figura 2.9: Aceleración (*speed-up*) obtenida usando la **Opción IV** para 240 cuadros de las secuencias *Basketball Pass* (BP), *Race Horses* (RH) y *Four People* (FP).



	2P	4P	6P	8P	10P	12P
Opt. I	1.60	2.79	4.31	6.08	6.89	7.78
Opt. II	1.93	3.72	5.47	7.69	8.71	9.98
Opt. III	1.97	3.86	5.76	7.26	9.33	10.87
Opt. IV	1.93	3.95	5.74	7.48	9.24	10.61

Figura 2.10: Aceleración (*speed-up*) obtenida para 240 cuadros de la secuencia *BQ Terrace*, usando las cuatro propuestas de paralelización.

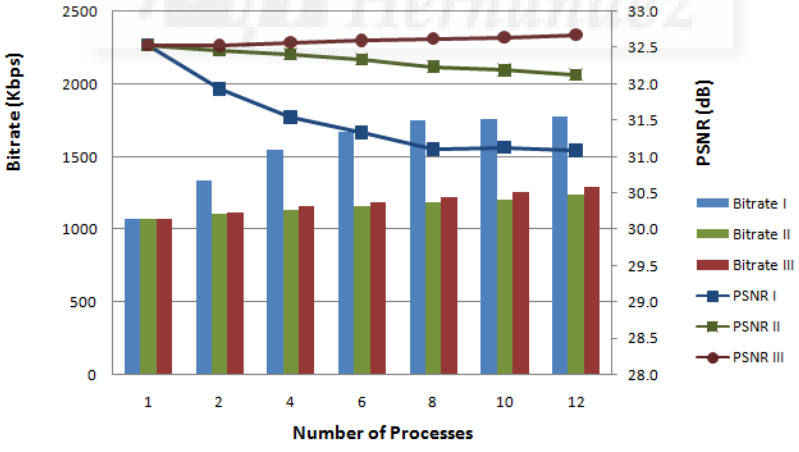


Figura 2.11: Distorsión (PSNR) y tasa de *bits* obtenida para 240 cuadros de la secuencia *Race Horses*, usando distinto número de procesos, para las Opciones I, II y III.

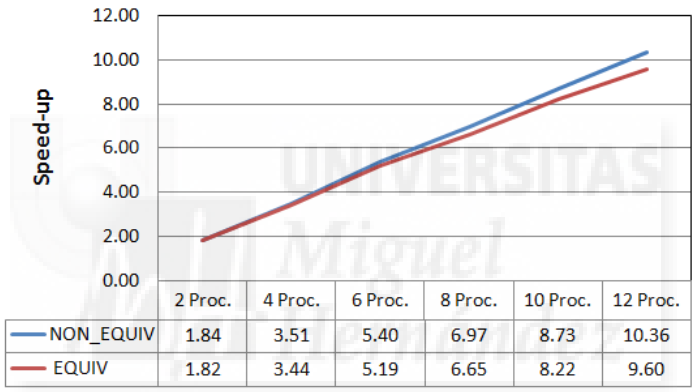


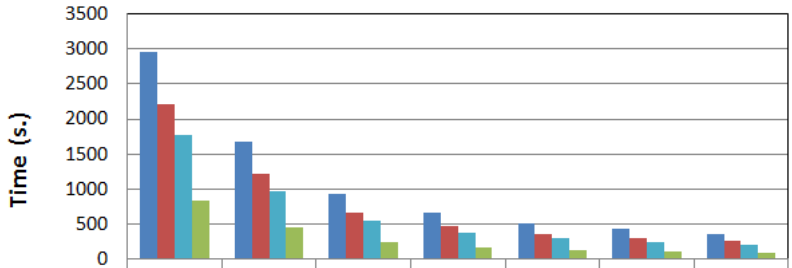
Figura 2.12: Aceleración (*speed-up*) obtenida para 480 cuadros de la secuencia *Basketball Pass*, usando la Opción III, generando vídeos comprimidos equivalentes y no equivalentes.

2.2.2. Paralelización a nivel de *slice*

La propuesta de paralelización a nivel de *slice* consiste en dividir cada cuadro en un número de *slices* igual al número de procesos en paralelo y asignar a cada proceso la codificación de una *slice*. En los modos de codificación LP, LB y RA, los cuadros de tipo P y B usan otros cuadros como referencia para realizar la estimación y compensación de movimiento. Para poder usar un cuadro como referencia se debe esperar a que todas las *slices* de dichos cuadros hayan sido codificadas. Por lo tanto todos los procesos en paralelo deben sincronizarse antes de codificar un cuadro. Este mecanismo de sincronización hace que la paralelización a nivel de *slice* sea menos eficiente computacionalmente que la paralelización a nivel de GOP, ya que los procesos, cuando terminan de codificar su *slice*, tienen que hacer una espera no ocupada hasta que el proceso más lento termine de codificar su *slice*. Para intentar minimizar los tiempos muertos de los procesos se utiliza un particionado homogéneo de cada cuadro, buscando que todas las *slices* sean de un tamaño lo más similar posible. Aún así, el hecho de que las *slices* sean del mismo tamaño no garantiza que los procesos tarden el mismo tiempo en codificar su *slice* ya que, debido al contenido diferente en cada una de ellas, el tiempo empleado para su codificación es variable. En el modo de codificación AI, donde todos los cuadros son de tipo I, no sería necesario aplicar el mecanismo de sincronización para codificar el vídeo pero se ha mantenido el mismo esquema por homogeneidad. Además, como se puede comprobar en los resultados, la variabilidad en el tiempo de codificación de las *slices* de un cuadro de tipo I es menor que para las *slices* de los cuadros de tipo P y B, con lo cual, la penalización es menor.

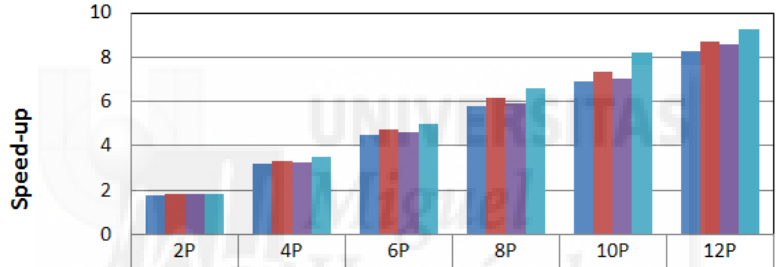
Para la obtención de resultados, además de usar distintas secuencias, se ha variado el valor del parámetro de cuantización (QP - *Quantization Parameter*) de HEVC. La cuantización es el proceso en el que se agrupan coeficientes de forma que el número de valores distintos de dichos coeficientes se reduce y por tanto se utilizan menos *bits* para representar la información. Este proceso es el que provoca que la compresión de vídeo sea una compresión con pérdidas y la calidad del vídeo reconstruído no sea exactamente igual a la del vídeo original. En la práctica, un valor de QP bajo implica una tasa menor de compresión y por lo tanto una mayor calidad del vídeo reconstruído. Un valor de QP alto implica una alta tasa de compresión y una menor calidad en el vídeo reconstruído. Dependiendo de las restricciones de ancho de banda de la red y de la calidad mínima requerida en nuestra aplicación podemos variar el parámetro QP a la hora de codificar un vídeo para ajustarlo a nuestras necesidades.

En la Figura 2.13 se presentan el tiempo de computación y la aceleración,



	1P	2P	4P	6P	8P	10P	12P
LowDelay-B	2964.7	1672.9	937.1	661.9	512.3	430.3	358.1
Random	2219.8	1215.4	668.7	467.8	361.0	302.5	255.8
LowDelay-P	1781.0	972.6	545.9	388.0	301.0	252.8	207.1
Intra	842.4	458.7	242.9	169.5	128.2	102.7	91.3

(a) Tiempo de computación

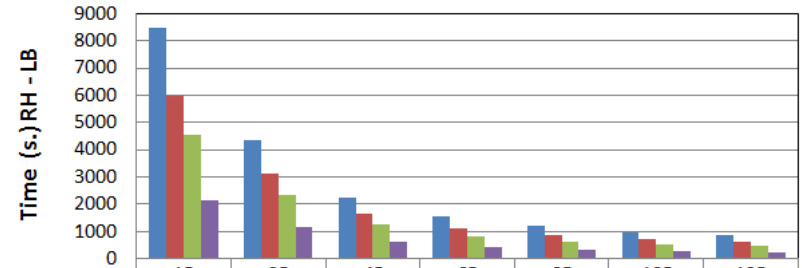


	2P	4P	6P	8P	10P	12P
LowDelay-B	1.8	3.2	4.5	5.8	6.9	8.3
Random	1.8	3.3	4.7	6.1	7.3	8.7
LowDelay-P	1.8	3.3	4.6	5.9	7.0	8.6
Intra	1.8	3.5	5.0	6.6	8.2	9.3

(b) Aceleración

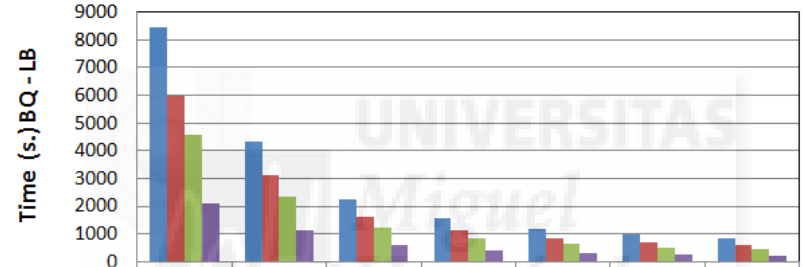
Figura 2.13: Tiempos de computación (a) y aceleración (*speed-up*) (b), para 120 cuadros de la secuencia *Four People* codificada con los modos AI, LB, LP y RA, con el parámetro QP=32, para distinto número de procesos.

procesando 120 cuadros de la secuencia *Four People* y usando los modos AI, LP, LB y RA, para distinto número de procesos y un valor de QP de 32. El algoritmo paralelo obtiene aceleraciones de 9,3 para 12 procesos en el modo AI y la media de la eficiencia es de 0,8. Se observa que el modo AI obtiene mejores valores de eficiencia que los modos LP, LB y RA. Esto es debido, como se ha indicado anteriormente, a que estos tres modos producen valores muy diferentes en los resíduos y el tiempo empleado para realizar la codificación entrópica difiere de una *slice* a otra. La Figura 2.14 muestra el



	1P	2P	4P	6P	8P	10P	12P
LowDelay-B	8467.8	4341.9	2259.2	1553.2	1184.3	971.6	836.5
Random	5958.1	3142.4	1627.3	1108.2	843.1	690.4	603.0
LowDelay-P	4555.1	2351.7	1229.3	832.2	634.7	515.6	456.2
Intra	2114.7	1135.8	612.9	423.0	304.2	250.4	218.1

(a) *Race Horses*



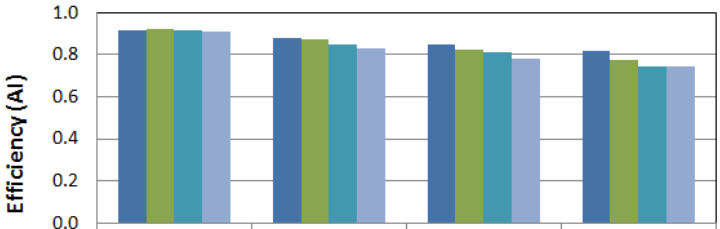
	1P	2P	4P	6P	8P	10P	12P
LowDelay-B	8467.8	4341.9	2259.2	1553.2	1184.3	971.6	836.5
Random	5958.1	3142.4	1627.3	1108.2	843.1	690.4	603.0
LowDelay-P	4555.1	2351.7	1229.3	832.2	634.7	515.6	456.2
Intra	2114.7	1135.8	612.9	423.0	304.2	250.4	218.1

(b) *BQ Terrace*

Figura 2.14: Tiempos de computación para 120 cuadros de las secuencias *Race Horses* y *BQ Terrace* codificadas con los modos AI, LB, LP y RA, con el parámetro QP=32, para distinto número de procesos.

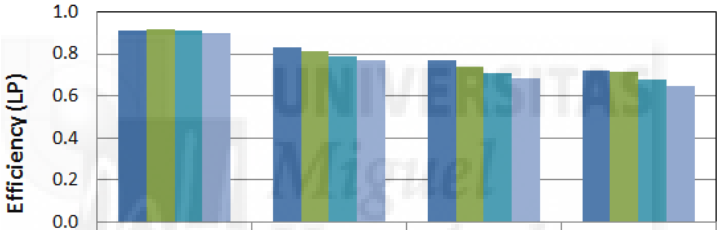
tiempo de computación para las secuencias *Race Horses* y *BQ Terrace* con un valor de QP de 32. Si miramos a la Figura 2.13 y a la Figura 2.14 se observa que el comportamiento computacional es similar para las tres secuencias. Este comportamiento se extiende para otros valores de QP.

En las Figuras 2.15, 2.16, 2.17 y 2.18 se presenta la comparación de eficiencias en función del número de procesos y del parámetro QP. Se obtienen buenos valores de eficiencia para todos los modos. Cuando el número



	2P	4P	8P	12P
QP=37	0.92	0.88	0.85	0.82
QP=32	0.92	0.87	0.82	0.77
QP=27	0.92	0.85	0.81	0.74
QP=22	0.91	0.83	0.78	0.74

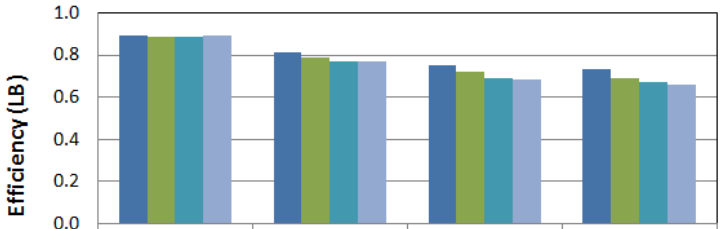
Figura 2.15: Eficiencia en el modo **All Intra** para 120 cuadros de la secuencia *Four People* codificada para distinto número de procesos y valores de QP.



	2P	4P	8P	12P
QP=37	0.91	0.83	0.77	0.72
QP=32	0.92	0.82	0.74	0.72
QP=27	0.91	0.79	0.71	0.68
QP=22	0.90	0.77	0.68	0.65

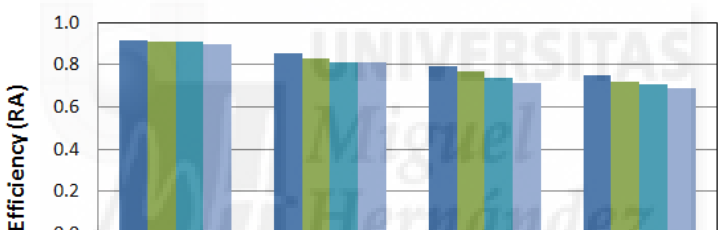
Figura 2.16: Eficiencia en el modo **Low-Delay P** para 120 cuadros de la secuencia *Four People* codificada para distinto número de procesos y valores de QP.

de procesos aumenta, la eficiencia disminuye. Aún así, la pérdida de eficiencia por este motivo es relativamente baja. La pérdida de eficiencia es menor para valores altos de QP (alto nivel de compresión) y se debe a que con altos niveles de compresión el tamaño de los datos residuales tiende a igualarse y por lo tanto se igualan los tiempos de compresión de las *slices*, lo que reduce el tiempo de espera no ocupada. De media se han obtenido unas eficiencias de 0,75, 0,78, 0,79 y 0,83 para los modos LB, LP, RA y AI, respectivamente. En la Figura 2.19 están representadas las eficiencias para las secuencias *Race Horses* y *BQ Terrace* codificadas con un valor de QP de 32. Se observa que los



	2P	4P	8P	12P
QP=37	0.89	0.81	0.75	0.73
QP=32	0.89	0.79	0.72	0.69
QP=27	0.88	0.77	0.69	0.67
QP=22	0.89	0.77	0.69	0.66

Figura 2.17: Eficiencia en el modo **Low-Delay B** para 120 cuadros de la secuencia *Four People* codificada para distinto número de procesos y valores de QP.

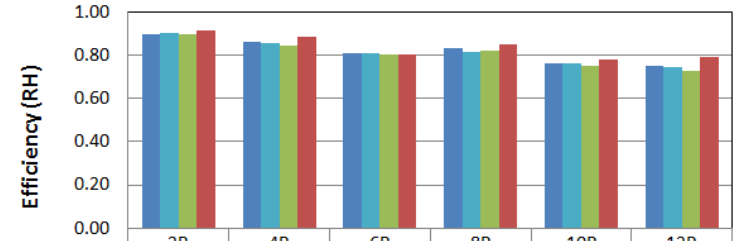


	2P	4P	8P	12P
QP=37	0.92	0.85	0.80	0.75
QP=32	0.91	0.83	0.77	0.72
QP=27	0.91	0.81	0.74	0.71
QP=22	0.90	0.81	0.71	0.69

Figura 2.18: Eficiencia en el modo **Random Access** para 120 cuadros de la secuencia *Four People* codificada para distinto número de procesos y valores de QP.

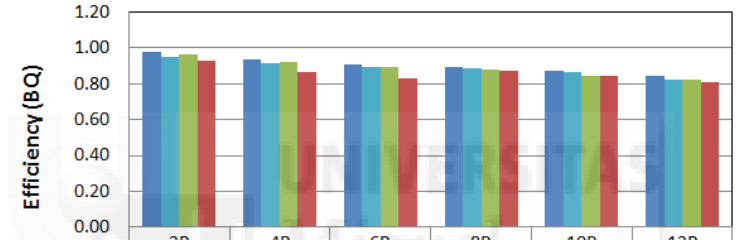
resultados son similares a los obtenidos para la secuencia *Four People*.

En las Figuras 2.20, 2.21, 2.22 y 2.23 se muestra el incremento en la tasa de *bits* en función del número de procesos y del parámetro QP. En los cuatro modos de codificación se incrementa la tasa de *bits* cuando aumenta el número de *slices* por cuadro y también cuando aumenta el valor del parámetro QP (aumento de la tasa de compresión). En general, para los cuatro modos, cuando aumenta el valor de QP, el tamaño de la parte residual del vídeo



	2P	4P	6P	8P	10P	12P
LowDelay-B	0.90	0.86	0.81	0.83	0.76	0.75
Random	0.91	0.86	0.81	0.82	0.76	0.75
LowDelay-P	0.90	0.85	0.80	0.82	0.75	0.73
Intra	0.92	0.89	0.80	0.85	0.78	0.79

(a) *Race Horses*

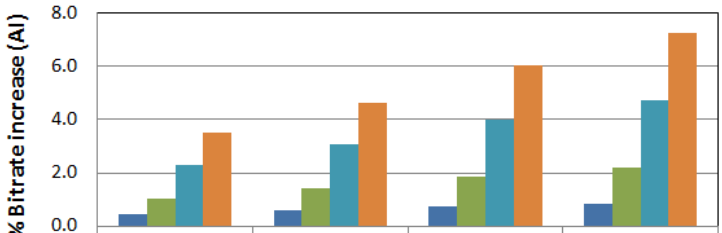


	2P	4P	6P	8P	10P	12P
LowDelay-B	0.98	0.94	0.91	0.89	0.87	0.84
Random	0.95	0.92	0.90	0.88	0.86	0.82
LowDelay-P	0.96	0.92	0.89	0.88	0.84	0.82
Intra	0.93	0.86	0.83	0.87	0.84	0.81

(b) *BQ Terrace*

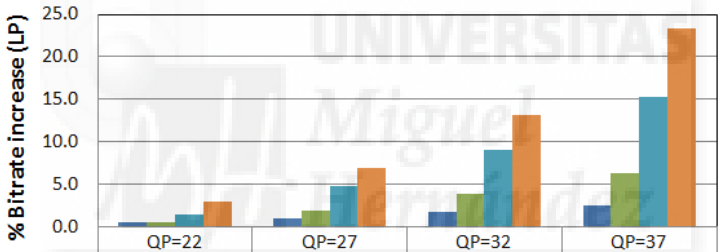
Figura 2.19: Eficiencia en los modos **All Intra**, **Low-Delay P**, **Low-Delay B** y **Random Access**, para 120 cuadros de las secuencias *Race Horses* y *BQ Terrace* codificadas para distinto número de procesos con un valores de QP de 32.

codificado disminuye, con lo que los *bits* introducidos por las cabeceras de las *slices* producen un incremento porcentual mayor. Para el modo AI (Figura 2.20), además del incremento causado por las cabeceras de las *slices*, se produce un incremento en la tasa de *bits* debido a que la predicción espacial está limitada a la región de cada *slice* y cuando se incrementa el número de *slices* por cuadro estas regiones son menores y la eficiencia en la codificación disminuye. En el modo AI, el máximo incremento en la tasa de *bits* es del 7,28 % y se produce al codificar con 12 *slices* por cuadro y con un valor de QP de 37. El incremento medio en tasa de *bits* es menor del 3 %. En los otros tres modos (Figuras 2.21, 2.22 y 2.23), además del incremento en la tasa de *bits*



	QP=22	QP=27	QP=32	QP=37
2P	0.42	0.58	0.72	0.83
4P	1.03	1.40	1.87	2.20
8P	2.31	3.10	3.98	4.72
12P	3.49	4.61	6.03	7.28

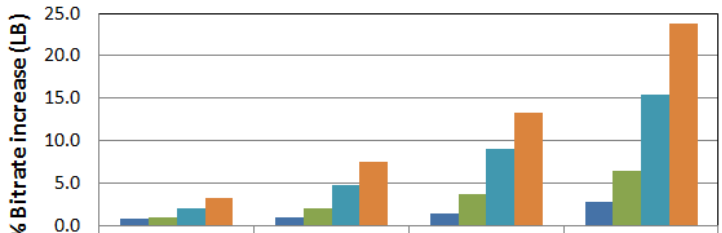
Figura 2.20: Evolución del incremento de la tasa de *bits* (%) en el modo **All Intra** para 120 cuadros de la secuencia *Four People* codificada para distinto número de procesos y valores de QP.



	QP=22	QP=27	QP=32	QP=37
2P	0.55	0.92	1.74	2.53
4P	0.41	1.85	3.86	6.22
8P	1.44	4.72	9.08	15.30
12P	2.88	6.93	13.16	23.40

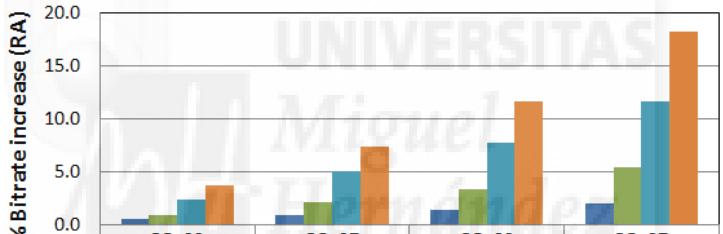
Figura 2.21: Evolución del incremento de la tasa de *bits* (%) en el modo **Low-Delay P** para 120 cuadros de la secuencia *Four People* codificada para distinto número de procesos y valores de QP.

producido por las cabeceras de las *slices*, no se pueden usar las predicciones para los vectores de movimiento más allá de los límites de la *slice* y además el codificador entrópico no puede utilizar el conocimiento de las regiones cercanas si pertenecen a otra *slice*. Todas estas penalizaciones en la compresión llevan a un incremento máximo en la tasa de *bits* del 23,81 %, 23,40 % y 18,15 % para los modos LB, LP y RA, respectivamente. En la Figura 2.24 se muestra el incremento en tasa de *bits* para el modo RA y las



	QP=22	QP=27	QP=32	QP=37
2P	0.77	0.91	1.41	2.81
4P	0.92	2.02	3.70	6.38
8P	1.94	4.79	9.08	15.41
12P	3.23	7.49	13.25	23.81

Figura 2.22: Evolución del incremento de la tasa de *bits* (%) en el modo **Low-Delay B** para 120 cuadros de la secuencia *Four People* codificada para distinto número de procesos y valores de QP.

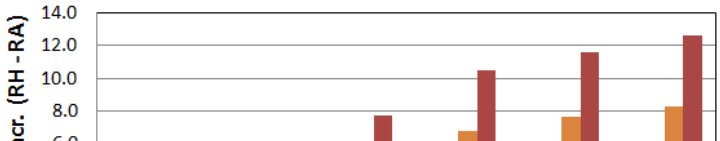


	QP=22	QP=27	QP=32	QP=37
2P	0.45	0.90	1.35	1.97
4P	0.89	2.14	3.36	5.33
8P	2.37	5.01	7.73	11.55
12P	3.63	7.31	11.63	18.15

Figura 2.23: Evolución del incremento de la tasa de *bits* (%) en el modo **Random Access** para 120 cuadros de la secuencia *Four People* codificada para distinto número de procesos y valores de QP.

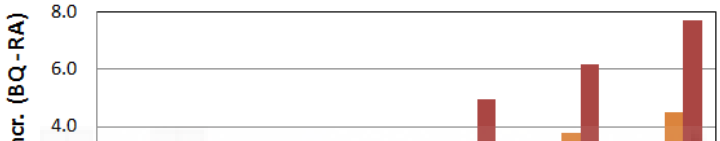
secuencias *Race Horses* y *BQ Terrace*.

En cuanto a la calidad del vídeo reconstruído, en la Figuras 2.25, 2.26, 2.27 y 2.28 se muestra la evolución de la tasa de *bits* (barras verticales) y de la distorsión (curvas horizontales), de la secuencia *Four People* para distinto número de procesos y valores de QP. En la Figura 2.29 se observa el mismo tipo de gráfica para las secuencias *Race Horses* y *BQ Terrace* codificadas en modo LP. Se puede observar que para todas las secuencias, modos y número de



	2P	4P	6P	8P	10P	12P
QP=22	0.37	1.20	1.88	2.42	2.73	2.83
QP=27	0.62	2.07	3.44	4.41	4.95	5.30
QP=32	0.99	3.07	5.28	6.80	7.62	8.24
QP=37	1.51	4.68	7.69	10.47	11.62	12.66

(a) *Race Horses*



	2P	4P	6P	8P	10P	12P
QP=22	0.12	0.32	0.49	0.61	0.85	0.95
QP=27	0.35	0.66	1.21	1.65	2.06	2.43
QP=32	0.58	1.45	2.20	2.96	3.77	4.50
QP=37	0.74	2.07	3.45	4.97	6.18	7.71

(b) *BQ Terrace*

Figura 2.24: Evolución del incremento de la tasa de *bits* (%) en el modo **Random Access** para 120 cuadros de las secuencias *Race Horses* y *BQ Terrace* codificadas para distinto número de procesos y valores de QP.

procesos, las diferencias en la calidad del vídeo reconstruído son mínimas con respecto a la versión secuencial (1 *slice* por cuadro).

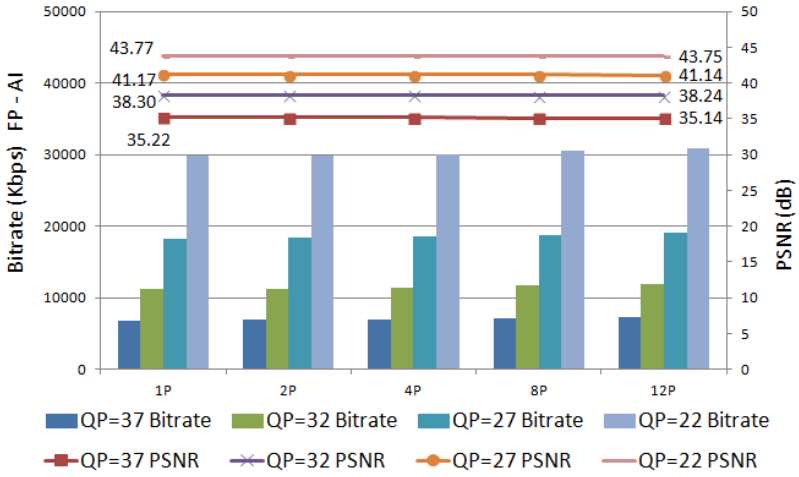


Figura 2.25: Distorsión (PSNR) y tasa de *bits* obtenidas en el modo **All Intra** para 120 cuadros de la secuencia *Four People* codificada para distinto número de procesos y valores de QP.

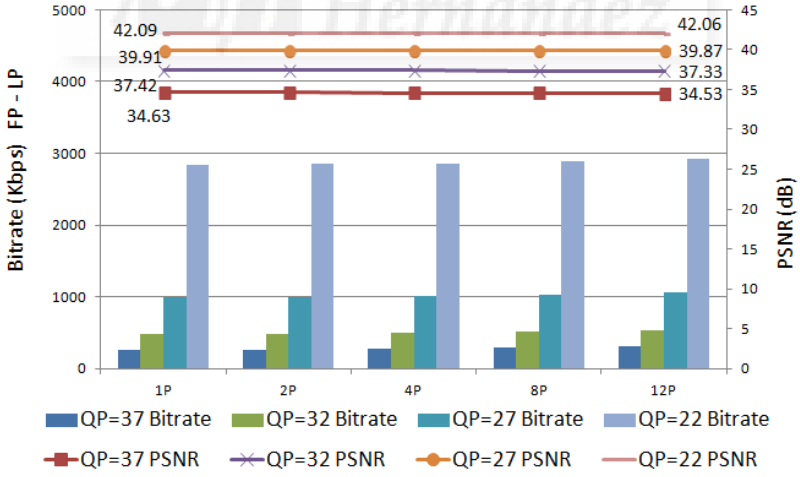


Figura 2.26: Distorsión (PSNR) y tasa de *bits* obtenidas en el modo **Low-Delay P** para 120 cuadros de la secuencia *Four People* codificada para distinto número de procesos y valores de QP.

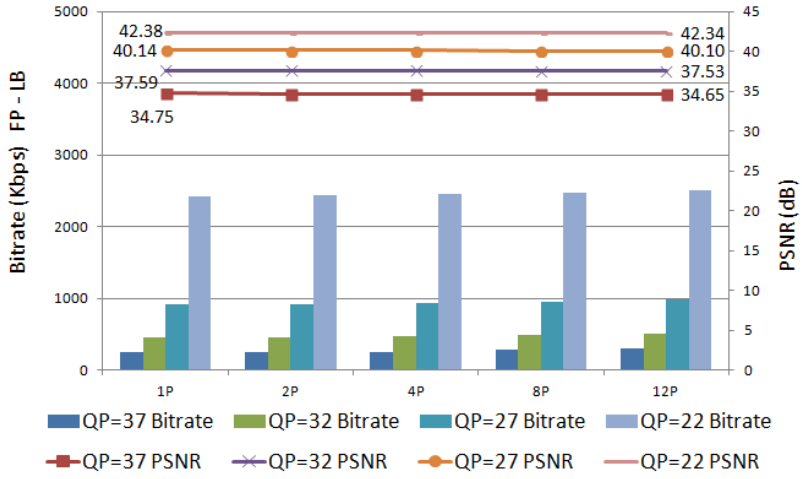


Figura 2.27: Distorsión (PSNR) y tasa de *bits* obtenidas en el modo **Low-Delay B** para 120 cuadros de la secuencia *Four People* codificada para distinto número de procesos y valores de QP.

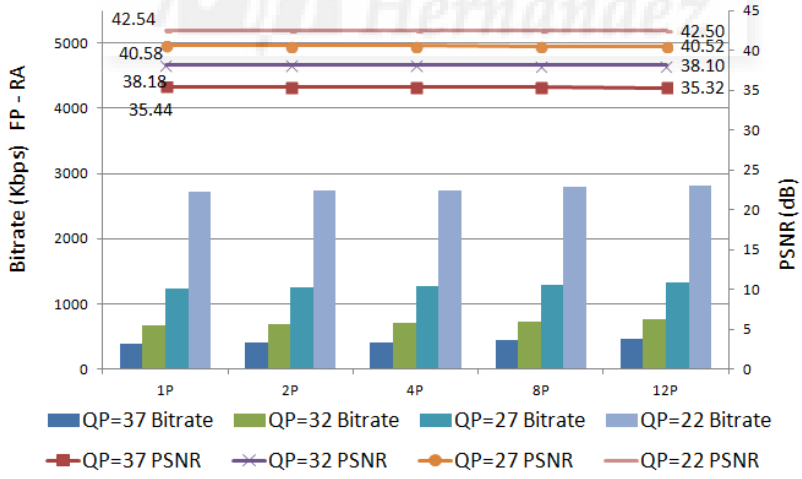
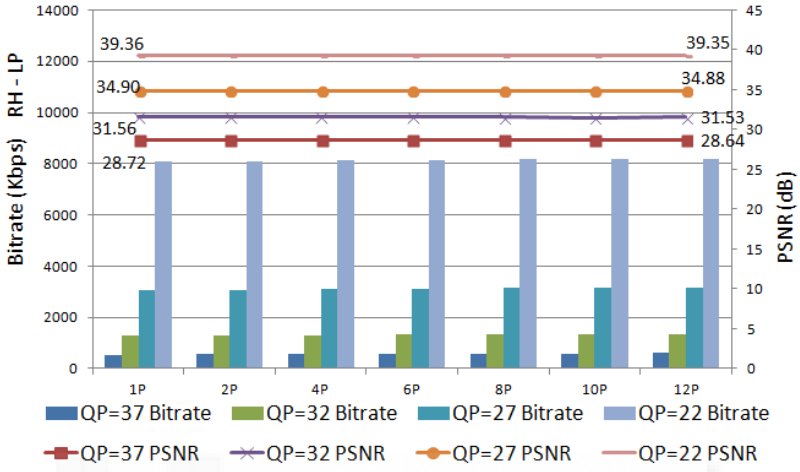
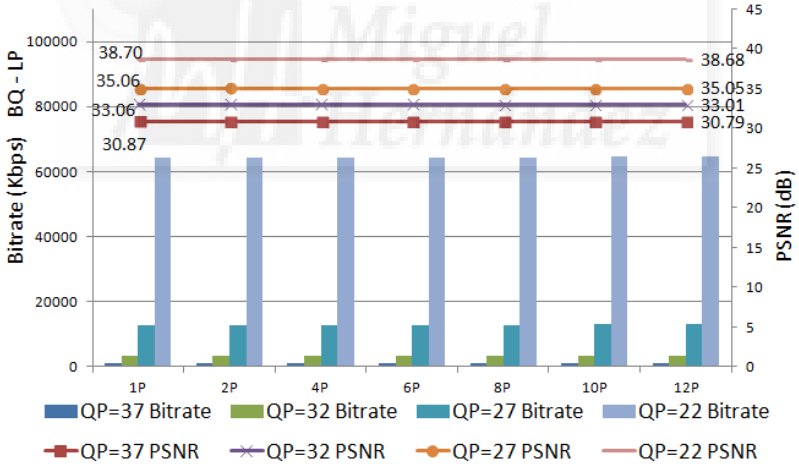


Figura 2.28: Distorsión (PSNR) y tasa de *bits* obtenidas en el modo **Random Access** para 120 cuadros de la secuencia *Four People* codificada para distinto número de procesos y valores de QP.



(a) *Race Horses*



(b) *BQ Terrace*

Figura 2.29: Distorsión (PSNR) y tasa de *bits* obtenidas en el modo **Low-Delay P** para 120 cuadros de las secuencias *Race Horses* y *BQ Terrace* codificadas para distinto número de procesos y valores de QP.

2.2.3. Protección a nivel de *slice*

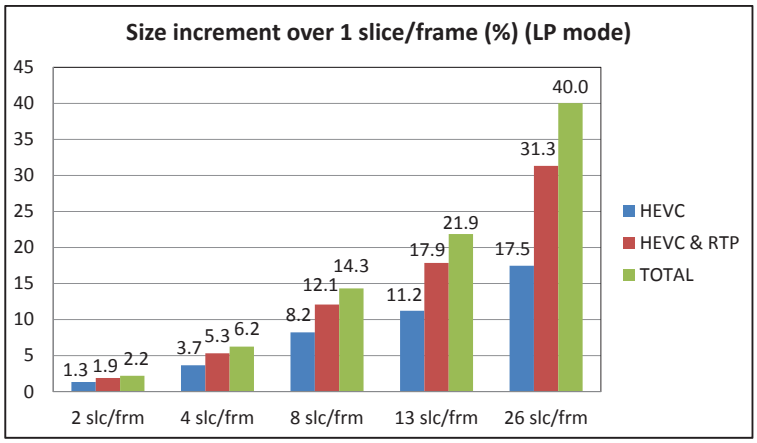
Como se ha explicado en el capítulo de introducción, las *slices* fueron diseñadas con el principal objetivo de proporcionar a los codificadores de vídeo un cierto nivel de resistencia ante errores. Su característica principal, el hecho de que puedan ser decodificadas de forma independiente del resto de *slices* del mismo cuadro, hace que la pérdida o los errores en la transmisión de una *slice* no afecte al resto de *slices* de dicho cuadro. Además, variando el número de *slices* por cuadro se puede ajustar el tamaño de cada *slice* codificada para que quepa íntegramente en un paquete de red. En ese caso, el porcentaje de paquetes de red perdidos coincidirá exactamente con el porcentaje de *slices* perdidas. Por el contrario, si tenemos *slices* que son mucho mayores que el valor de MTU entonces aumenta la vulnerabilidad del vídeo codificado ante la pérdida de paquetes. Si cada *slice*, por ejemplo, es 10 veces mayor que el valor de MTU, entonces necesitaremos unos 10 paquetes de red para transportar cada *slice*. De esta forma, el número de paquetes de red perdidos no necesariamente implicará el mismo número de *slices* perdidos. Si se pierde, por ejemplo, un 1 % de los paquetes de red y cada paquete de red perdido corresponde a una *slice* diferente, entonces el porcentaje de *slices* perdidas (y por tanto de información perdida para la decodificación del vídeo) no será del 1 % sino que aumentará hasta el 10 %. A continuación se muestra el estudio de los resultados de dividir cada cuadro en distinto número de *slices*. Por una parte se evalúa el rendimiento en cuanto a la eficiencia de compresión y por otra parte se analizan los datos que afectan al transporte del vídeo en una red vehicular, como son, la proporción de paquetes de red por cada *slice* y la tasa de paquetes por segundo necesaria para la transmisión del vídeo. También se propone un mecanismo de protección de los paquetes a nivel de red y se evalúa dicha propuesta en un entorno de red vehicular realista (basado en un mapa real).

Para la investigación se ha utilizado la secuencia *Race Horses* y dos modos de codificación: AI y LP. En el modo LP se ha modificado el fichero de configuración original de forma que cada 32 cuadros se introduzca uno de tipo I. Si no se hiciera esta modificación y usáramos en las pruebas el modo LP original (que codifica el primer cuadro de tipo I y el resto de cuadros de tipo P), al producirse una pérdida de datos en un cuadro, el resto de la secuencia de vídeo quedaría totalmente inservible. Esto se debe a que los cuadros de tipo P se codifican usando la estimación y compensación de movimiento. Un error en un cuadro afectará también a los cuadros de tipo P que lo usan como referencia, ya que, aunque esos cuadros se reciban correctamente, como están basados en el cuadro que tiene el error, la decodificación producirá cuadros erróneos. Estos cuadros a su vez serán referencia de otros cuadros de tipo P,

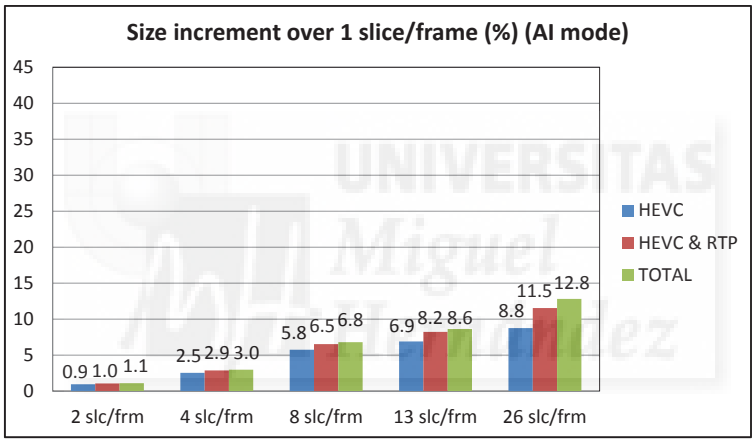
con lo que el error se irá extendiendo hasta el final de la secuencia. Si añadimos un cuadro I cada cierto número de cuadros P, el decodificador puede restablecer los errores y reajustar la decodificación ya que un cuadro I no usa otros cuadros como referencia, eliminando el "efecto dominó" de las pérdidas en cuadros anteriores. A este mecanismo se le denomina "refresco intra". Para ambos modos de codificación se ha elegido un valor de QP de 37, que produce un valor de PSNR de 32,12 dB para el modo AI (1 *slice* por cuadro) y un valor de PSNR de 30,19 dB para el modo LP (1 *slice* por cuadro).

En la Figura 2.30 se muestra el incremento porcentual en el tamaño de la secuencia de vídeo codificada para diversos números de *slices* por cuadro con respecto al tamaño para una única *slice* por cuadro. Por una parte se muestra el porcentaje de incremento del flujo de *bits* codificado y también el porcentaje de incremento al añadirle la cabecera RTP a cada *slice*. Además se ha añadido otra cabecera que nos permite tener el control sobre las *slices* fragmentadas, cuyo tamaño sea superior al valor de MTU. La Figura 2.30(a) muestra los resultados usando el modo LP y la Figura 2.30(b) muestra los resultados para el modo AI. Los valores etiquetados como "HEVC" indican el incremento porcentual en el tamaño del vídeo codificado (sin incluir las cabeceras RTP). Este incremento es debido a los *bits* extra de las cabeceras de las *slices* y a la pérdida de eficiencia en el método de compresión, causada por la imposibilidad de usar la predicción más allá de los límites de cada *slice*. Los valores etiquetados como "HEVC & RTP" muestran el incremento producido incluyendo la cabecera RTP. La curva con los valores "Total" tiene también en cuenta la cabecera de fragmentación. Como el tamaño de las *slices* en el modo LP es mucho menor que las del modo AI, entonces el porcentaje de incremento en el tamaño del vídeo codificado es mucho mayor (el tamaño de las cabeceras que se añaden a cada *slice* en los cuadros de tipo P es relativamente alto con respecto a los datos que incluyen). En los peores casos, en el modo LP se aumenta el tamaño del vídeo codificado en un 20 % al codificarlo con 13 *slices* por cuadro y un 40 % si lo codificamos con 26 *slices* por cuadro. En cambio, para el modo AI y los mismos casos, los incrementos son de sólamente el 8,6 % y el 12,8 % respectivamente.

En la Figura 2.31, la curva etiquetada como "Without FEC" muestra los valores absolutos de tasa de *bits* de la secuencia para los modos LP y AI. En el modo LP la tasa de *bits* varía desde 64,9 kbps (para 1 *slice* por cuadro) hasta 90,9 kbps (para 26 *slices* por cuadro). En el modo AI el rango va desde 324,4 kbps (1 *slice* por cuadro) hasta 365,9 kbps (26 *slices* por cuadro). Como se ha dicho antes, la tasa de *bits* generada en el modo LP es mucho menor que la generada en el modo AI, concretamente, cuando se codifica con un *slice* por cuadro, la tasa de *bits* del modo LP es 5 veces menor que la del modo AI. El resto de curvas de esta figura se interpretará más adelante cuando se explique la protección a nivel de red propuesta.



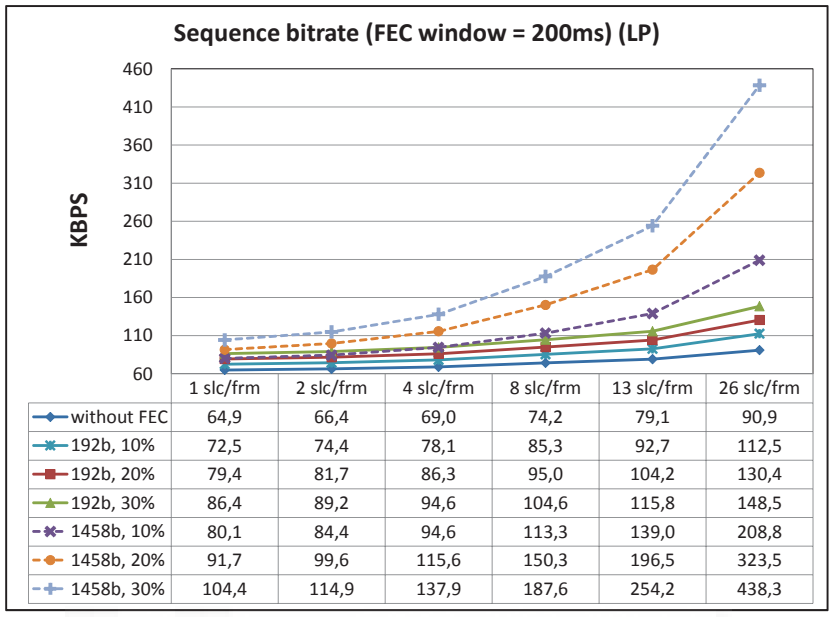
(a) Low-Delay P



(b) All Intra

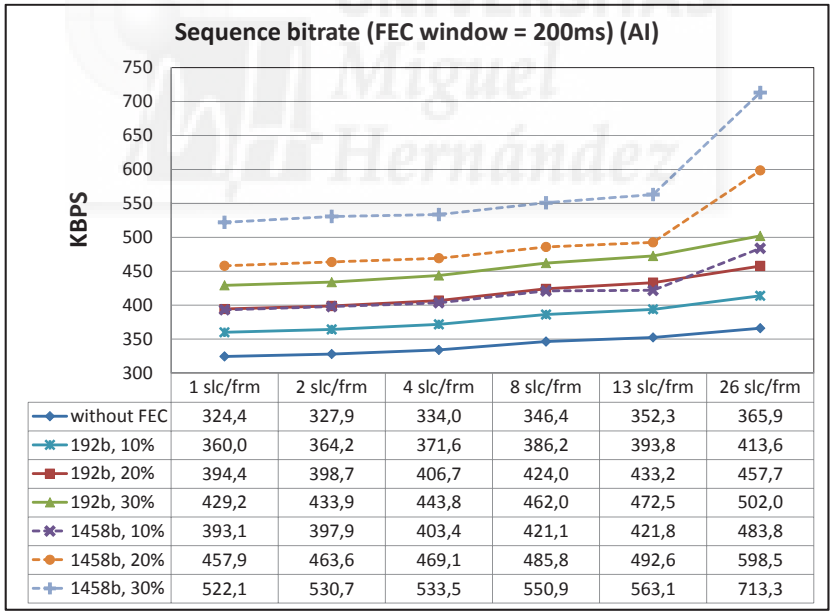
Figura 2.30: Porcentaje de incremento en la tasa de *bits* para distinto número de *slices* por cuadro. *(HEVC, HEVC + cabecera RTP y HEVC + cabecera RTP + cabecera de fragmentación).*

Un factor de relevancia a la hora de proteger los datos en la transmisión del vídeo es la proporción de paquetes de red (fragmentos) respecto a los paquetes RTP (cada paquete RTP incluye una *slice*). Cuando la proporción tiende a 1 entonces la pérdida de un paquete de red sólo afectará a un paquete RTP. Además, si se usa un mecanismo de protección de paquetes (como se mostrará más adelante), la recuperación de cada paquete de red perdido se traducirá directamente en una mejora de la calidad de vídeo reconstruído. La proporción media de fragmentos por paquete RTP se muestra en la Tabla 2.2. Los datos muestran que, cuando se usa el modo AI y un número bajo de *slices*



	1 slc/frm	2 slc/frm	4 slc/frm	8 slc/frm	13 slc/frm	26 slc/frm
without FEC	64,9	66,4	69,0	74,2	79,1	90,9
192b, 10%	72,5	74,4	78,1	85,3	92,7	112,5
192b, 20%	79,4	81,7	86,3	95,0	104,2	130,4
192b, 30%	86,4	89,2	94,6	104,6	115,8	148,5
1458b, 10%	80,1	84,4	94,6	113,3	139,0	208,8
1458b, 20%	91,7	99,6	115,6	150,3	196,5	323,5
1458b, 30%	104,4	114,9	137,9	187,6	254,2	438,3

(a) Low-Delay P



	1 slc/frm	2 slc/frm	4 slc/frm	8 slc/frm	13 slc/frm	26 slc/frm
without FEC	324,4	327,9	334,0	346,4	352,3	365,9
192b, 10%	360,0	364,2	371,6	386,2	393,8	413,6
192b, 20%	394,4	398,7	406,7	424,0	433,2	457,7
192b, 30%	429,2	433,9	443,8	462,0	472,5	502,0
1458b, 10%	393,1	397,9	403,4	421,1	421,8	483,8
1458b, 20%	457,9	463,6	469,1	485,8	492,6	598,5
1458b, 30%	522,1	530,7	533,5	550,9	563,1	713,3

(b) All Intra

Figura 2.31: Tasa de *bits*, sin protección FEC y con protección FEC usando una ventana temporal de protección de 200 ms, para distintos tamaños de símbolo de protección, porcentaje de redundancia y número de *slices* por cuadro (incluyendo las cabeceras RTP y de fragmentación).

Tabla 2.2: Proporción media de fragmentos (paquetes de red) por cada paquete RTP (*slice*).

fragmentos/RTP	1 sl	2 sl	4 sl	8 sl	13 sl	26 sl
Modo LP	1.87	1.37	1.21	1.02	1.00	1.00
Modo AI	7.90	4.29	2.38	1.51	1.07	1.00

por cuadro, la proporción se aleja de 1, y esto indica que la recuperación de paquetes no será tan productiva como debería.

Como se ha dicho anteriormente, para la protección de los paquetes de red se ha usado un mecanismo FEC. Se han probado diferentes configuraciones para observar su comportamiento y para evaluar cuáles de ellas son más apropiadas. Cada una de las secuencias codificadas en los dos modos y para distinto número de *slices* por cuadro ha sido protegida usando códigos RaptorQ. Se han utilizado 6 tamaños diferentes para la ventana temporal de protección (133, 200, 250, 333, 500 y 1000 ms). Se han asignado 3 niveles diferentes de protección (10 %, 20 % y 30 %). Se han usado 2 combinaciones de tamaño de símbolos de redundancia y de número de paquetes de recuperación (192 *bytes* con 7 paquetes de reparación y 1458 *bytes* con 1 paquete de reparación). En primer lugar se analiza el incremento generado pora la protección añadida con los códigos RaptorQ. Además del incremento en tasa de *bits* se ha medido el incremento en el número de paquetes por segundo que se han de transmitir (hay que tener en cuenta que al añadir los paquetes de protección esta tasa aumenta). En trabajos previos, se observó que en las redes vehiculares la tasa de paquetes por segundo a transmitir influye más en las pérdidas que el tamaño de cada paquete en sí. En la Tabla 2.3 se muestra el número de paquetes por segundo en la transmisión de la secuencia codificada con el modo LP, tanto sin añadir los paquetes FEC, como incluyendo la protección al 30 % para distintos tamaños de símbolos de redundancia y todos los tamaños de ventana temporal de protección. Cuando se usa un tamaño de símbolo grande se generan más paquetes por segundo que cuando se usa un tamaño pequeño. Esta relación empeora cuando aumentamos el número de *slices* por cuadro. También se observa que variar la ventana temporal de protección no modifica significativamente la tasa de paquetes por segundo.

La Tabla 2.4 muestra la tasa de paquetes por segundo sin protección FEC y usando protección FEC con un paquete de protección con un tamaño fijo de 192 *bytes* y una ventana temporal de 200 ms. Si se compara el número de

Tabla 2.3: Tasa de paquetes por segundo en el modo LP, sin protección FEC y con protección FEC al 30 % de redundancia, para diferentes tamaños de símbolos de protección, diferentes ventanas temporales de protección y diferente número de *slices* por cuadro.

Paquetes/Seg.	1 sl	2 sl	4 sl	8 sl	13 sl	26 sl
SIN FEC	56.1	82.0	134.6	244.9	391.3	780.0
192 b,133 ms	75.4	102.7	156.2	270.0	420.7	824.6
192 b,200 ms	73.8	100.5	155.0	268.7	419.6	822.7
192 b,250 ms	73.1	100.1	154.6	268.4	419.0	821.5
192 b,333 ms	72.9	99.8	154.2	267.7	418.4	820.7
192 b,500 ms	72.1	99.2	153.3	266.5	417.0	821.5
192 b,1000 ms	71.2	97.9	152.0	265.2	415.5	820.9
1458 b,133 ms	83.2	116.3	183.0	320.4	509.1	1016.7
1458 b,200 ms	82.9	114.9	181.3	321.7	509.8	1015.1
1458 b,250 ms	81.8	114.5	180.8	321.6	509.1	1010.0
1458 b,333 ms	81.8	114.0	180.7	320.0	507.9	1008.9
1458 b,500 ms	80.5	113.7	179.1	317.8	504.3	1003.9
1458 b,1000 ms	79.5	111.6	176.2	313.2	498.1	1013.9

paquetes por segundo para los modos LP y AI usando la protección FEC, se puede observar la correlación con la Tabla 2.2. Cuando se codifica la secuencia con 26 *slices* por cuadro tenemos una proporción de 1 fragmento por paquete RTP tanto para el modo AI como para el modo LP. Esto indica que cada paquete RTP es menor que el valor de MTU y la tasa de paquetes por segundo se calcula multiplicando directamente la tasa de cuadros por segundo (30 cuadros/segundo) por el número de *slices* por cuadro, en este caso 780,0 pps (paquetes por segundo). En el lado opuesto de la tabla Tabla 2.4, si nos fijamos en el modo AI usando 1 *slice* por cuadro y sin usar protección FEC se observa que la tasa de paquetes por segundo (237,0 pps) está muy lejos de multiplicar los cuadros por segundo por el número de *slices* por cuadro.

En la Figura 2.31 vemos la tasa de *bits* para las secuencias codificadas en los modos AI y LP sin protección FEC y con protección FEC, para los 3 niveles de redundancia y los 2 tamaños de símbolo de protección. Seleccionar un tamaño de 192 *bytes* conduce a unas tasas de *bits* mucho menores. Este resultado aparece mucho más enfatizado para el modo de codificación LP cuando se codifica usando un número elevado de *slices* por cuadro.

El caso de estudio de la red vehicular está basado en el escenario que se

Tabla 2.4: Tasa de paquetes por segundo, sin protección FEC y con protección FEC con un tamaño de símbolo de protección de 192 *bytes*, y una ventana de protección de 200 ms para los dos modos de codificación, diferentes porcentajes de redundancia y diferente número de *slices* por cuadro.

Paquetes/Seg.	1 sl	2 sl	4 sl	8 sl	13 sl	26 sl
(LP) SIN FEC	56.1	82.0	134.6	244.9	391.3	780.0
(LP) 10%	63.6	89.9	143.3	254.9	402.0	795.4
(LP) 20%	68.5	95.0	148.9	261.9	410.8	808.9
(LP) 30%	73.8	100.5	155.0	268.7	419.6	822.7
(AI) SIN FEC	237.0	257.3	285.3	362.8	417.6	780.0
(AI) 10%	265.0	285.6	314.8	392.9	449.3	815.0
(AI) 20%	290.2	311.0	340.6	421.1	478.2	847.6
(AI) 30%	316.0	336.8	367.7	449.4	507.4	880.7

muestra en la Figura 2.32. Corresponde a un área de 2000m *x* 2000m de la ciudad de Kiev. En la Figura 2.33 podemos ver el mapa seleccionado una vez han sido convertidos sus datos para poder ser manejados con el simulador SUMO. En la Figura 2.34 se representa la interfaz gráfica de usuario del simulador OMNeT++/MiXiM/Veins donde hemos realizado las evaluaciones de la transmisión de vídeo en la red vehicular. En dicho escenario se han colocado 3 RSUs (marcadas como A, B y C) a lo largo de una avenida de aproximadamente 2000m. Las tres antenas transmiten simultáneamente el vídeo codificado y protegido con los códigos RaptorQ. Los vehículos que actúan como clientes del vídeo transmitido, seleccionan el canal de servicio por el que transmiten las antenas y van recogiendo los paquetes con el vídeo codificado y protegido. Con estos paquetes se intenta recuperar los paquetes perdidos usando la información redundante de los códigos RaptorQ. Las *slices* (paquetes RTP) para las que se haya recibido o recuperado todos los fragmentos que la componen podrán ser utilizados en la decodificación del vídeo y las *slices* a las que les falte al menos uno de los fragmentos que la componen no se podrán usar. El vídeo codificado, sin las *slices* no recuperadas se decodifica con el decodificador de HEVC (que ha sido modificado) y posteriormente se calcula el valor PSNR para el vídeo reconstruído. Para evaluar las pérdidas de paquetes se han realizado experimentos con 4 tasas de tráfico de fondo: 0, 30, 240 y 390 paquetes por segundo (512 *bytes*). Se han realizado simulaciones usando los dos modos de codificación (AI y LP), diferente número de *slices* por cuadro (1, 4, 8 y 13 *slices*/cuadro) y tres niveles de protección (10 %, 20 % y 30 %). En las zonas con buena cobertura, las

Figura 2.32: Vista parcial de la ciudad de Kiev. El cuadrado indica el área seleccionada para lanzar las simulaciones.

pérdidas de paquetes son debidas al tráfico de fondo. Aquí encontramos pérdidas aisladas. Por el contrario, en las zonas cercanas a los límites de cobertura de las RSUs las pérdidas de paquetes tienen estructura de ráfagas, porque la señal es muy débil o incluso se llega a perder. En las pérdidas aisladas los códigos RaptorQ hacen un buen trabajo recuperando paquetes perdidos. En las Tablas 2.5 y 2.6 se muestran los resultados de las simulaciones, usando una redundancia del 30 % y con una tasa de tráfico de fondo de 390 paquetes por segundo, en zonas con buena cobertura, para los modos de codificación LP y AI, respectivamente. Se muestra el porcentaje de paquetes de red perdidos (*Pérdida TOTAL*), el porcentaje de paquetes RTP que no son útiles (perdidos completamente o con algún fragmento perdido) después de haber realizado la recuperación por medio de los códigos RaptorQ (*Pérdida RTP*) y la diferencia en calidad respecto del vídeo sin pérdidas (*Pérdida PSNR*).

Como se puede comprobar, los códigos RaptorQ pueden recuperar, en general, un alto porcentaje de paquetes perdidos y el resultado es una pérdida de paquetes RTP reducida. Aunque esto no ocurre cuando se usa el modo AI y un número pequeño (1 y 4) de *slices* por cuadro. En estas configuraciones el número de fragmentos por paquete RTP es mayor y esto penaliza la recuperación de paquetes RTP. En el modo AI con 1 *slice* por cuadro y un

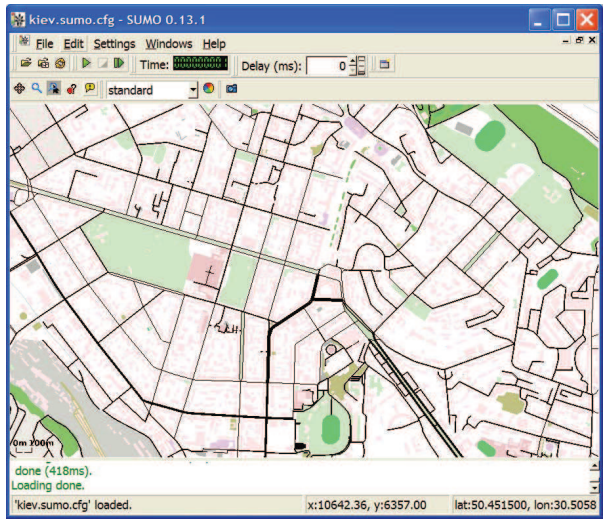


Figura 2.33: Representación en el simulador SUMO de los datos obtenidos de OpenStreetMap (una vez realizado el proceso de transformación).

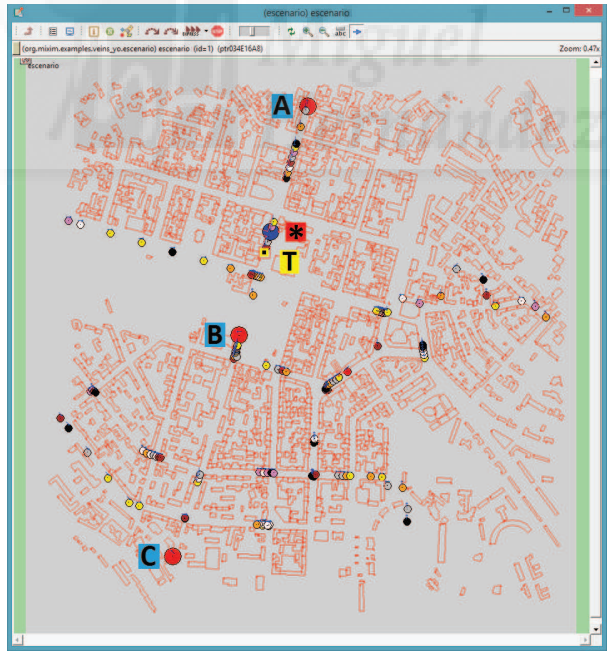


Figura 2.34: Escenario de red vehicular en OMNeT++/MiXiM/Veins. *(A,B,C = RSUs // * = cliente de vídeo // T = fuente de tráfico de fondo // círculos pequeños = otros vehículos // polígonos rojos = edificios).*

Tabla 2.5: Porcentaje de pérdidas totales de paquetes, porcentaje de paquetes RTP perdidos después de la recuperación FEC, y pérdida de calidad en PSNR del vídeo reconstruído, para una tasa de tráfico de fondo de 390 pps y un nivel de protección del 30 %, para áreas con una buena cobertura. **Modo LP**.

slices/cuadro	medida	192 b	1458 b
1 slc/cuadro	Pérdida TOTAL (%)	11.28	10.89
1 slc/cuadro	Pérdida RTP (%)	1.28	0.18
1 slc/cuadro	Pérdida PSNR (dB)	0.99	0.18
4 slc/cuadro	Pérdida TOTAL (%)	13.55	14.36
4 slc/cuadro	Pérdida RTP (%)	1.84	0.62
4 slc/cuadro	Pérdida PSNR (dB)	2.85	0.75
8 slc/cuadro	Pérdida TOTAL (%)	14.59	15.05
8 slc/cuadro	Pérdida RTP (%)	1.47	0.09
8 slc/cuadro	Pérdida PSNR (dB)	1.90	0.11
13 slc/cuadro	Pérdida TOTAL (%)	13.97	13.25
13 slc/cuadro	Pérdida RTP (%)	0.29	0.00
13 slc/cuadro	Pérdida PSNR (dB)	0.52	0.00

tamaño de símbolo de 192 *bytes* se tiene el 19,07 % de pérdidas de paquetes de red y después de la recuperación las pérdidas quedan en 14,81 % de paquetes RTP. En cambio para el mismo modo de codificación y con 13 *slices* por cuadro las pérdidas de paquetes de red (15,06 %) producen un resultado de sólamente el 0,46 % de paquetes RTP perdidos. En la tabla 2.2 se encuentra la explicación numérica a este comportamiento. Comparando las Tablas 2.5 y 2.6 se observa que el modo AI es inherentemente más resistente ante los errores que el modo LP. Vemos, por ejemplo, que una pérdida del 16,94 % de paquetes RTP en el modo AI produce una pérdida de calidad de 2,52 dB y que en el modo LP, con sólo una pérdida del 1,84 % de paquetes RTP se produce una pérdida en calidad de 2,85 dB. Tal y como se explicó anteriormente, el "efecto dominó" en el modo LP lo hace mucho más vulnerable. Otro hecho observable en estas dos tablas es que cuando se elige un tamaño de símbolo de 1458 *bytes* las propiedades de recuperación son siempre (salvo un caso aislado) mejores que cuando se elige un tamaño de símbolo de 192 *bytes*. A pesar de este comportamiento, es más recomendable usar un tamaño de símbolo pequeño si atendemos a la penalización en el incremento de tasa de *bits* (Figura 2.31) y de paquetes por segundo (Tabla 2.3) que introduce un tamaño grande de símbolo.

Tabla 2.6: Porcentaje de pérdidas totales de paquetes, porcentaje de paquetes RTP perdidos después de la recuperación FEC, y pérdida de calidad en PSNR del vídeo reconstruído, para una tasa de tráfico de fondo de 390 pps y un nivel de protección del 30 %, para áreas con una buena cobertura. **Modo AI**.

slices/cuadro	medida	192 b	1458 b
1 slc/cuadro	Pérdida TOTAL (%)	19.07	19.10
1 slc/cuadro	Pérdida RTP (%)	14.81	16.94
1 slc/cuadro	Pérdida PSNR (dB)	2.20	2.52
4 slc/cuadro	Pérdida TOTAL (%)	18.34	18.25
4 slc/cuadro	Pérdida RTP (%)	6.76	4.17
4 slc/cuadro	Pérdida PSNR (dB)	1.32	0.81
8 slc/cuadro	Pérdida TOTAL (%)	16.65	16.13
8 slc/cuadro	Pérdida RTP (%)	2.56	0.93
8 slc/cuadro	Pérdida PSNR (dB)	0.71	0.23
13 slc/cuadro	Pérdida TOTAL (%)	15.06	14.72
13 slc/cuadro	Pérdida RTP (%)	0.46	0.21
13 slc/cuadro	Pérdida PSNR (dB)	0.15	0.06

Capítulo 3

Conclusiones y trabajo futuro

Índice

3.1. Conclusiones

En esta tesis se ha abordado el problema de la transmisión de vídeo en redes vehiculares. Los dos principales retos a superar son el gran ancho de banda requerido y la degradación en la calidad debido a la pérdida de paquetes de datos durante la transmisión. La reducción en el ancho de banda necesario para la transmisión se puede acometer mediante la utilización de los codificadores más recientes, que permiten doblar la eficiencia en la codificación, reduciendo casi a la mitad el flujo de *bits* para niveles similares de calidad. El inconveniente de usar estos codificadores es que introducen una complejidad computacional elevada. Para solucionar este problema se han diseñado varias estrategias de paralelización para el codificador HEVC, tanto a nivel de GOP como a nivel de *slice*. Las evaluaciones realizadas muestran resultados muy favorables en cuanto al nivel de aceleración de la codificación del vídeo. El otro obstáculo a superar es la pérdida de paquetes. Se han propuesto y evaluado distintas técnicas para preservar la calidad del vídeo transmitido en las redes vehiculares. La utilización de las *slices* permite paliar el efecto que producen los paquetes perdidos. La combinación de la utilización de las *slices* así como de técnicas de protección de paquetes a nivel de red permiten dotar de robustez a la transmisión de vídeo en las redes vehiculares.

En las 4 propuestas de paralelización a nivel de GOP, los esquemas diseñados consiguen aceleraciones casi óptimas. La estrategia diseñada para ser usada con el modo de codificación AI se encuentra siempre entre las más eficientes computacionalmente y además produce exactamente el mismo flujo de *bits* y la misma calidad que la versión secuencial. De media consigue eficiencias de aceleración del 91,34 % y al generar exactamente el mismo flujo de *bits* no introduce una disminución en la calidad del vídeo. Las otras tres propuestas a nivel de GOP están diseñadas para el modo LP. En estos tres esquemas se modifica la estructura de procesamiento de los GOPs y esto introduce una penalización en la eficiencia en la codificación respecto del algoritmo secuencial original. La Opción III es la más eficiente de las tres y consigue, de media, eficiencias de aceleración del 94,65 %, y aunque, como se ha dicho, en dicha versión se incurre en penalizaciones en el flujo de *bits*, es la que menos distorsión introduce de las tres alternativas. Por último, se han realizado modificaciones al fichero de configuración del modo LP para que el algoritmo secuencial genere el mismo flujo de *bits* que los algoritmos paralelos. En dicho caso sólo se reduce de media un 3,62 % de eficiencia en aceleración y se obtiene el mismo nivel de calidad.

La propuesta de aceleración a nivel de *slice* se basa en dividir cada cuadro en un número de *slices* igual al número de procesos en paralelo. La división de

los cuadros en *slices* introduce un incremento en el flujo de *bits*, con respecto a realizar la codificación usando una única *slice* por cuadro. Como cada *slice* es decodificable de forma independiente, la división en *slices* permite paralelizar el procesamiento del vídeo, además de permitir introducir propuestas de protección ante errores. Mediante la introducción de las *slices* atacamos directamente los dos retos planteados, mejorar la velocidad de codificación y mejorar la resistencia ante las pérdidas de datos. Este esquema de paralelización obtiene resultados más discretos, aunque interesantes, en cuanto a la eficiencia en aceleración de la codificación del vídeo. En concreto para los modos AI, LP, LB y RA obtiene, de media, eficiencias del 83 %, 78 %, 75 % y 79 %, respectivamente. La penalización media en el tamaño del flujo de *bits* para el modo AI es del 3 % y para los modos LP, LB y RA es del 6 %. En estos tres modos (LP, LB y RA) para un alto número de *slices* por cuadro y tasas muy altas de compresión, el incremento porcentual en el flujo de *bits* no es aceptable, pero en el resto de casos este esquema de paralelización permitirá aumentar el rendimiento computacional con la utilización de las *slices*, cuyo uso se hace imprescindible si queremos paliar el efecto de las pérdidas.

La protección del vídeo ante errores de transmisión se ha centrado alrededor de la utilización de las *slices*. Al ser elementos sintácticos del vídeo comprimido que son decodificables de forma independiente, se puede ajustar el tamaño de las *slices* para que las pérdidas de los paquetes de red impliquen la pérdida de la mínima cantidad de información útil. Esto se consigue si la proporción de paquetes de red por cada paquete RTP tiende a 1. Para proteger los paquetes de red se han utilizado los códigos RaptorQ. Se ha evaluado el rendimiento de la protección del vídeo por medio de simulaciones vehiculares realistas, usando mapas reales. El modo de codificación AI se muestra inherentemente más resistente a errores que el modo LP ya que se evita el efecto dominó, debido a que cada cuadro se decodifica de forma independiente al resto de cuadros de la secuencia. Para evitar en cierto modo el citado efecto dominó en el modo de codificación LP se ha introducido el refresco intra a nivel de cuadro. Se ha estudiado cómo influyen las *slices* sobre la eficiencia de codificación y también sobre dos factores que influyen en la vulnerabilidad o resistencia ante las pérdidas: la proporción de paquetes de red por cada paquete RTP y la tasa de paquetes de red por segundo necesarios para la transmisión del vídeo. En el modo AI, cuando se usa un número pequeño de *slices* por cuadro, se producen muchos fragmentos por cada paquete RTP, lo que hace que, a pesar de ser más resistente ante errores, su vulnerabilidad aumente y las recuperaciones de paquetes de red perdidos no siempre incrementen la calidad en el vídeo recuperado. A su vez, el modo LP ajusta mejor el tamaño de los paquetes RTP al valor de MTU lo que hace que

prácticamente la totalidad de los paquetes recuperados puedan ser utilizados en la decodificación del vídeo. Dependiendo de si los cuadros de referencia usados por dichos paquetes recuperados son correctos o erróneos, entonces la calidad del vídeo decodificado mejorará o no. En cuanto a la elección del tamaño de símbolo de protección para la utilización de los códigos RaptorQ se ha observado que elegir un tamaño grande de símbolo proporciona mejores resultados en cuanto a la recuperación de paquetes perdidos pero genera un incremento no aceptable en el flujo de *bits*. En cambio, la elección de un tamaño de símbolo de recuperación pequeño, produce unos buenos resultados en la recuperación de paquetes manteniendo el incremento en el flujo de *bits* controlado. Mediante las simulaciones se constata que en las pérdidas por ráfagas, los códigos RaptorQ no pueden llevar a cabo con éxito la recuperación de los paquetes ya que no hay un número mínimo de paquetes correctos recibidos. En cambio se ha comprobado el buen comportamiento de los códigos RaptorQ ante las pérdidas aisladas. Como conclusión general se ha observado que la elección de un número medio de *slices* por cuadro lleva a las condiciones óptimas en cuanto a la robustez en la transmisión del vídeo en redes vehiculares. En nuestras evaluaciones, en las que hemos utilizado 1, 2, 4, 8, 13 y 26 *slices* por cuadro la opción que utiliza 8 *slices* por cuadro se ha mostrado la más eficiente.

3.2. Trabajo futuro

Como trabajos futuros que permitan ampliar el presente trabajo de investigación se proponen:

- Crear un modelo de paralelización híbrido que permita reunir las dos principales propuestas de este trabajo de forma conjunta, combinando la paralelización a nivel de GOP con la paralelización a nivel de *slice*.

- Extender la investigación sobre el procesamiento en paralelo aplicado al codificador HEVC con otras aproximaciones como la paralelización a nivel de *tile* y la técnica WPP (*Wavefront Parallel Processing*).

- Estudiar la protección de HEVC usando los *tiles* como unidad sintáctica y comparar su rendimiento respecto a las *slices*.

- Introducir el refresco intra a nivel de *slice* y comparar su rendimiento respecto al refresco intra a nivel de cuadro.

3.3. Otras publicaciones

A continuación se enumeran otras publicaciones en las que ha participado el autor, relacionadas con la codificación de vídeo y su transmisión en redes inalámbricas.

- A. Torres, P. Pinol, C.T. Calafate, J.C. Cano, and P. Manzoni, "Evaluating H.265 Real-Time Video Flooding Quality in Highway V2V Environments", Wireless Communications and Networking Conference (WCNC), pages 2716-2721, April 2014.

- Martinez Rach, M.O., Piñol, P.J., Lopez Granado, O.M., Malumbres, M.P., Oliver, J., Calafate, C., "On the Performance of Video Quality Assessment Metrics under Different Compression and Packet Loss Scenarios", The Scientific World Journal, 2014.

- Piñol, P.J., Torres, A., Lopez Granado, O.M., Martinez Rach, M.O., Malumbres, M.P., "Evaluating HEVC Video Delivery in VANET Scenarios", IFIP Wireless Days, Valencia, 2013.

- Torres A., Piñol, P.J., Calafate, C.T., Cano, J.C., Manzoni, P., "V2V Real-time Video Flooding on Highways", XXIV Jornadas de Paralelismo, JP2013, Madrid, 2013.

- Piñol, P.J., Torres, A., Lopez Granado, O.M., Martinez Rach, M.O., Malumbres, M.P., "An Evaluation of HEVC using Common Conditions", XXIV Jornadas de Paralelismo, JP2013, Madrid, 2013.

- Pablo Piñol, Otoniel López, Miguel Martínez, José Oliver, Manuel P. Malumbres, "Modeling video Streaming over VANETs", 7th ACM Workshop on Performance Monitoring and Measurement of Heterogeneous Wireless and Wired Networks, PM2HW2N'12, pages 7-14, New York, 2012.

- Piñol, P.J., Martinez Rach, M.O., Lopez Granado, O.M., Oliver, J., Malumbres, M.P., "Video Transmission Simulations in Vehicular Adhoc Networks", XXIII Jornadas de Paralelismo, JP2012, Elche, 2012.

- O. López, P. Piñol, M. Martínez-Rach, M.P. Malumbres and J. Oliver, "Low-Complexity 3D-DWT Video Encoder Applicable to IPTV", Signal Processing: Image Communication (ISSN: 0923-5965), vol. 26, n. 7, pp. 358-369, 2011.

- Otoniel M. Lopez, Miguel O. Martinez-Rach, Pablo Piñol, Jose Oliver, Manuel Perez Malumbres, “Rate Control Algorihtms for Non-embedded Wavelet-Based Image Coding”, Journal of Signal Processing Systems - Springer, 2011.

- Pablo Piñol, Otoniel Lopez, Miguel Martinez-Rach, Manuel P. Malumbres, José Oliver, Carlos T. Calafate, “Testing Performance of Current Video Codecs in Teleoperated Mobile Robot Applications: A Practical Experience”, Mobile Robots Navigation, 1st Edition, p-501-514, Intech, ISBN: 978-953-307-076-6, 2010.

- Oliver, J., Lopez Granado, O.M., Martinez Rach, M.O., Piñol, P.J., Calafate, C., Malumbres, M.P., “International Standards for Image Compression”, Encyclopedia of Information Science and Technology, Second Edition p-2164-2169 ISBN: 978-1-60566-026-4, 2008.

- Piñol, P.J., Martinez Rach, M.O., Lopez Granado, O.M., Malumbres, M.P., Oliver, J., “A Practical Study of Video Streaming Over IEEE 802.11 Wireless Networks using DirectShow Video Encoders”, XVIII Jornadas de Paralelismo, JP2007, Zaragoza, 2007.

- P. Piñol, M. Martinez Rach, O. Lopez, M.P. Malumbres, J. Oliver, “Analyzing the Impact of Commercial Video Encoders in Remotely Teleoperated Mobile Robots through IEEE 802.11 Wireless Network Technologies”, 5th IEEE International Conference on Industrial Informatics, INDIN2007, pages 425-430, Viena, 2007.

Apéndice I

Acrónimos

AI	All Intra
AVC	Advanced Video Coding
B	Bi-directional predictive
FEC	Forward Error Correction
GOP	Group Of Pictures
HEVC	High Efficiency Video Coding
OBU	On Board Unit
OpenMP	Open Multi-Processing
I	Intra
ITS	Intelligent Transportation System
JCT-VC	Joint Collaborative Team on Video Coding
LB	Low-delay B
LP	Low-delay P
MANET	Mobile Ad Hoc Network
MTU	Maximum Transfer Unit
OBU	On Board Unit
P	Predictive
PSNR	Peak Signal-to-Noise Ratio
QP	Quantization Parameter
RA	Random Access
RSU	Road Side Unit
RTP	Real-time Transport Protocol
SUMO	Simulation of Urban MObility
TraCI	TRAffic Control Interface
VANET	Vehicular Ad Hoc Network
WiMAX	Worldwide Interoperability for Microwave Access
WPP	Wavefront Parallel Processing

Apéndice II

Artículos

En cumplimiento de la normativa de la Universidad Miguel Hernández de Elche, en las siguientes páginas se recogen el conjunto de trabajos publicados.

- **Parallel strategies analysis over the HEVC encoder**

  P. Piñol, H. Migallón, O. López-Granado, and M. P. Malumbres

  Springer Science+Business Media
  The Journal of Supercomputing
  Volume 70, Issue 2, 13 pages
  http://dx.doi.org/10.1007/s11227-014-1121-1

- **Slice-based parallel approach for HEVC encoder**

  P. Piñol, H. Migallón, O. López-Granado, and M. P. Malumbres

  Springer Science+Business Media
  The Journal of Supercomputing
  Volume 71, Issue 5, 11 pages
  http://dx.doi.org/10.1007/s11227-014-1371-y

- **Protection of HEVC Video Delivery in Vehicular Networks with RaptorQ Codes**

  Pablo Piñol, Miguel Martínez-Rach, Otoniel López, and Manuel Pérez Malumbres

  Hindawi Publishing Corporation
  The Scientific World Journal
  Volume 2014, Article ID 619379, 9 pages
  http://dx.doi.org/10.1155/2014/619379

A continuación se especifica la clasificación de las revistas donde se han publicado los artículos, dentro del índice Journal Citation Reports de 2013.

- **The Journal of Supercomputing**

  Impact Factor: **0.841**
  Category Name: **COMPUTER SCIENCE, THEORY & METHODS**
  Total Journals in Category: **102**
  Journal Rank in Category: **47**
  Quartile in Category: **Q2**

- **The Scientific World Journal**

  Impact Factor: **1.219**
  Category Name: **MULTIDISCIPLINARY SCIENCES**
  Total Journals in Category: **55**
  Journal Rank in Category: **16**
  Quartile in Category: **Q2**

En cumplimiento de la normativa se indica la afiliación de los coautores de los artículos presentados en esta tesis. Todos ellos pertenecen al **Departamento de Física y Arquitectura de Computadores** de la **Universidad Miguel Hernández de Elche**, situada en la Avda. de la Universidad s/n, 03202, Elche, España.

- **Pablo José Piñol Peral**
  pablop@umh.es
- **Héctor Francisco Migallón Gomis**
  hmigallon@umh.es
- **Miguel Onofre Martínez Rach**
  mmrach@umh.es
- **Otoniel Mario López Granado**
  otoniel@umh.es
- **Manuel José Pérez Malumbres**
  mels@umh.es

J Supercomput (2014) 70:671–683
DOI 10.1007/s11227-014-1121-1

Parallel strategies analysis over the HEVC encoder

P. Piñol · H. Migallón · O. López-Granado · M. P. Malumbres

Published online: 2 March 2014

Abstract Recently, a new video coding standard called HEVC has been developed to deal with the nowadays media market challenges, being able to reduce to the half, on average, the bit stream size produced by the former video coding standard H.264/AVC at the same video quality. However, the computing requirements to encode video improving compression efficiency have significantly been increased. In this paper, we focus on applying parallel processing techniques to HEVC encoder to significantly reduce the computational power requirements without disturbing the coding efficiency. So, we propose several parallelization approaches to the HEVC encoder which are well suited to multicore architectures. Our proposals use OpenMP programming paradigm working at a coarse grain level parallelization which we call GOP-based level. GOP-based approaches encode simultaneously several groups of consecutive frames. Depending on how these GOPs are conformed and distributed, it is critical to obtain good parallel performance, taking also into account the level of coding efficiency degradation. The results show that near ideal efficiencies are obtained using up to 12 cores.

Keywords Parallel algorithms · Video coding · HEVC · Multicore · Performance

This research was supported by the Spanish Ministry of Education and Science under grant TIN2011-27543-C03-03, the Spanish Ministry of Science and Innovation under grant TIN2011-26254 and Generalitat Valenciana under grant ACOMP/2013/003.

P. Piñol · H. Migallón · O. López-Granado (✉) · M. P. Malumbres
Physics and Computer Architecture Department, Miguel Hernández University, 03202 Elche, Spain
e-mail: otoniel@umh.es

H. Migallón
e-mail: hmigallon@umh.es

1 Introduction

The new High Efficiency Video Coding (HEVC) standard has been recently developed by the Joint Collaborative Team on Video Coding (JCT-VC) which was established by the ISO/IEC Moving Picture Experts Group (MPEG) and ITU-T Video Coding Experts Group (VCEG). This new standard will replace the current H.264/AVC [1] standard to deal with nowadays and future multimedia market trends. 4K definition video content is a nowadays fact and 8K definition video will not take long to become a reality too. Furthermore, the new standard supports high quality color depth at 8 and 10 bits. The HEVC standard delivers the same video quality than the H.264/AVC high profile at half the bit rate.

Regarding complexity, HEVC decoder has a similar behavior to the H.264/AVC one [2]. However, HEVC encoder is several times more complex than H.264/AVC encoder and it will be a hot research topic in the years to come. At the time of developing this work, the current version of the reference software, called HEVC test model (HM), is HM 10.0 which corresponds to the HEVC text specification draft 10 [3]. A good overview of HEVC standard can be found in [4].

We can find in the literature several works about complexity analysis and parallelization strategies for the emerging HEVC standard as in [5–7]. Most of the available HEVC parallelization proposals are focused in the decoding side, looking for the most appropriate parallel optimizations at the decoder that provide real-time decoding of high-definition (HD) and ultra-high-definition (UHD) video contents. The most recent and efficient decoding algorithm is presented in [8] where authors present and approach wave parallel processing (WPP) called overlapped wavefront (OWF). In that approach, each coding tree block (CTB) row is decoded by a thread (WPP) and also the decoding execution of consecutive frames is overlapped using a restricted motion vector size. Results show that a speed-up of 7.6× is achieved when using 8 cores with a negligible loss in rate/distortion (R/D).

Currently, there are few works focused on the HEVC encoder. In [9], authors propose a fine-grain parallel optimization in the motion estimation module of the HEVC encoder allowing to perform the motion vector prediction in all prediction units (PUs) available at the coding unit (CU) at the same time. In [10], authors propose a parallelization inside the intra prediction module that consists on removing data dependencies among subblocks of a CU, obtaining interesting speed-up results. In [11], authors propose a parallel algorithm at CTU (coding tree unit) level for the intra coding mode. The algorithm, which has been implemented on ×265 [12], launches the intra prediction of each CTU on each thread achieving a speed-up up to 5× when using 7 cores.

In this paper, we will analyze the available parallel strategies in the HEVC standard and their viability over the HM reference software. Furthermore, we present a parallelization alternative for the HEVC encoder which is specially suited for low-delay encoding profiles. Our proposal works at group of pictures (GOP) processing level, following different parallel GOP-based strategies and analyzing the overall behavior in terms of complexity reduction and coding efficiency.

The remainder of this paper is organized as follows: in Sect. 2, an overview of the available profiles in HEVC and common test conditions are presented. Section 3 provides an overview of the high-level parallelism strategies proposed in the HEVC

standard. Section 4 presents the GOP-based parallel alternatives we propose for the low-delay application profile, while in Sect. 5a comparison between the proposed parallel alternatives is presented. Finally, in Sect. 6, some conclusions and future work are discussed.

2 HEVC profiles

In [13], the JCT-VC defines the common test conditions and software reference configurations to be used for HEVC experiments. In that paper, it can be found a series of settings to evaluate HEVC video codec and to compare the different contributions made to it.

A total of 24 video sequences are specified, arranged in 6 classes. Also, the quantization parameter (QP) and the set of configuration files for the encoding process are detailed. Using these common conditions makes easier to perform comparisons between innovative proposals. JCT-VC also provides a spreadsheet to calculate rate-distortion (RD) curves and the percentage of gain in bit rate, using Bjontegaard-Delta (BD) measurements [14].

Classes from A to E include natural video sequences at diverse frame sizes. Class F comprises sequences that contain synthetic video in part of them or in its whole. Two of the sequences in class A have a bit depth of 10 bits and the rest of the sequences have a bit depth of 8 bits. The frame rate of the sequences ranges from 20 to 60 fps.

Configuration files are provided within reference software package [15]. There are eight different test conditions which are a combination of 2 bit depths: Main (8 bits) and Main10 (10 bits) with 4 coding modes: All Intra (AI), Random Access (RA), Low-Delay B (LB), and Low-Delay P (LP).

In All Intra mode, every frame is coded as an I-frame i.e. it is coded without any motion estimation/compensation. So each frame is independent from the other frames in the sequence. This mode gets lower compression rates (compared to the other 3 modes) because P-frames and B-frames can usually obtain better compression rates than I-frames at the same quality level. On the other hand, the coding process for All Intra mode is faster than for the other 3 modes because no time is wasted in motion estimation. Every frame is coded in rendering order. Applications that require a fast encoding process and are not concerned about limited bandwidth or storage capacity, fit perfectly in this coding mode.

Random Access mode combines I-frames and B-frames in GOPs of 8 frames. A B-frame is a frame that uses motion estimation/compensation to achieve good compression rates. Each block of a B-frame can use up to 2 reference frames, so in the coding process 2 lists of reference pictures are maintained. Reference frames can be located earlier or later than the frame we are currently coding. In this mode, coding (and decoding) order is not the same as rendering order. To allow navigating along the coded sequence (pointing to a certain moment) or to allow functions like fast forward, an I-frame is inserted periodically. Depending on the frame rate of each sequence, the intra-refresh period varies. The intra period is a multiple of 8 (the size of the GOP) which inserts an I-frame approximately every second. Applications that do not have time constraints (when coding a video sequence) and need

features like the aforementioned fast forward, are the target applications of this coding mode.

Low-Delay modes (LP and LB) code each frame in rendering order. First, an I-frame is inserted in the coded bit stream and then only P-frames (or B-frames) are used for the rest of the complete video sequence. Here, 'P' stands for Predictive and 'B' stands for Bipredictive. This means that a P-frame uses only one previously encoded and decoded frame as reference for motion estimation and compensation and a B-frame uses a pair of frames to perform these tasks. In LB mode, this pair of frames is always selected from previous frames (in rendering order). On the contrary, RA mode uses past and future frames as reference for B-frames and this introduces a delay because you have to wait for future pictures to be available to encode/decode the present frame. In LP and LB modes GOP size is 4. This two modes achieve better compression performance than AI mode and do not suffer from the delay introduced by RA mode. Applications like video-conference which have bandwidth and time constraints can benefit from low-delay modes.

3 HEVC high-level parallelism strategies

High-level parallel strategies may be classified in a hierarchical scheme depending on the desired parallel grain size. So, we define from coarser to finer grain parallelism levels: GOP, tile, slice, and wavefront. When designing an HEVC parallel version, we first need to analyze the available hardware where the parallel encoder will run, to determine which parallelism levels are the most appropriate.

The coarsest parallelization level, GOP-based, is based on breaking the whole video sequence in GOPs in such a way that the processing of each GOP is completely independent from the other GOPs. In general, this approach will provide the best performance. However, depending on the way we define the GOPs structure and remove the inter-GOP dependencies, the coding performance may be affected.

Tiles are used to split a picture horizontally and vertically into multiple sub-pictures. Using tiles, prediction dependencies are broken just at tile boundaries. Consecutive tiles are represented in raster scan order. The scan order of coding tree blocks (CTBs) remains a raster scan. When splitting a picture horizontally, tiles may be used to reduce line buffer sizes in an encoder, as it operates on regions which are narrower than a full picture. Tiles also allow the composition of a picture from multiple rectangular sources that are encoded independently.

Slices follow the same concept as in H.264/AVC allowing a picture to be partitioned into groups of consecutive coding tree units (CTUs) in raster scan order. Each slice can be transmitted in a different network abstraction layer unit (NALU) that may be parsed and decoded independently, except for optional inter-slice filtering. There is a break in prediction dependences at slice boundaries, which causes a loss in coding efficiency. The use of slices is concerned with error resilience or with maximum transmission unit size matching but it has undoubtedly been exploited for parallel purposes in the past.

Wavefront parallel processing (WPP) technique splits a picture into CTU rows, where each CTU row may be processed by a different thread. Dependences between

rows are maintained except for the CABAC [16] context state, which is reinitialized at the beginning of each CTU row. To improve the compression efficiency, rather than performing a normal CABAC reinitialization, the context state is inherited from the second CTU of the previous row.

All high-level parallelization techniques become more useful with image sizes growing beyond HD for both encoder and decoder. At small frame sizes where real-time decoding in a single-threaded manner is possible, the overhead associated with parallelization removes any meaningful benefit. For large frame sizes, it might be useful to enforce a minimum number of picture partitions to guarantee a minimum level of parallelism for the decoder.

Current HM reference software does not directly support most of the high-level parallelism approaches mainly due to its implementation design. In the next section, we will present several GOP-based parallelization approaches that may be implemented in cluster-based or multicore-based hardware architectures.

4 Parallel algorithms

We have parallelized the HEVC reference software for LB and AI modes. This two modes are useful for applications that have time constraints, so we think they can benefit from parallelization strategies. Obviously, this work can be easily extended to use LP mode, but this is not true for RA mode due to the way it uses reference frames. In particular, RA mode uses both past and future frames as reference pictures so dependencies between frames are tighter than in the two evaluated modes.

The GOP-based parallel algorithms developed are suitable to run on shared memory platforms, distributed memory platforms and hybrid platforms. First of all, note that in AI mode the GOP size is 1, and, moreover, each frame is computed with no reference frames, therefore the GOP-based algorithm is inherently parallel, with the exception of the constraint of synchronization processes to generate the correct bit stream. In LB mode, the GOP size used is 4, however this value could be changed. Note that the GOP size is the minimum number of adjacent frames allocated to each process. As we have said, we have developed synchronous algorithms where the synchronization processes are performed after the GOP computations. At these synchronization points, each process writes data in the bit stream, obviously in an orderly way. In this work, we describe and analyze the four parallel approaches that we have developed. The first three options are applied to LB mode, and the last one is applied to AI mode, namely:

- Option I: (LB) in this option, we sequentially assign each GOP to one process in the parallel execution, so processes will encode isolated GOPs.
- Option II: (LB) in this approach, we divide the sequence in as many parts as the number of parallel processes, so that each process will encode a block of adjacent GOPs.
- Option III: (LB) similar to Option II, except that each process begins the coding by inserting an I-frame.
- Option IV: (AI) similar to Option I where each GOP is sequentially assigned to one process, but here a GOP consists of only one I-frame.

Figure 1 shows the parallel distribution performed when Option I is used. As we have said, the synchronization processes are located after GOP computations. The root process (Proc. 0 or P0) encodes the first frame as an I-frame and then sends it to the rest of threads so that they can use it as a reference picture. These threads are idle until the root process has computed this first frame.

Note that the I-frame encoding is very fast, therefore the parallel algorithm performance is not significantly affected. After that, all processes encode a GOP of 4 B-frames. All processes, except the root process, encode their first B-frame without a nearby reference picture; therefore, the number of bits needed to encode this first B-frame in those processes is greater than the number of bits needed to encode the first B-frame of P0.

The proposed parallel algorithms, are developed providing each process with its own working buffer, considering, on the one hand, to increase the performance of the developed algorithms, and on the other hand, to design suitable algorithms for shared memory platforms, distributed memory platforms and hybrid platforms. This fact changes the real pattern of the reference pictures used. For instance, in sequential processing, the second B-frame of a GOP uses frames $-1-2-6-10$ as reference pictures (-1 means the previous frame, and so on). As the GOP size is 4, frame -2 points to the last frame of the previous GOP (the frame two positions before the current frame in the original video sequence). In parallel processing, as we assign isolated GOPs to each process, the previous GOP is not the previous adjacent GOP in the original video sequence and therefore frame -2 will not point to the frame two positions before the current frame. If, for instance, the number of processes is 6, then the previous GOP for this process will be located in the video sequence 6 GOPs away from the current GOP. So for the second B-frame of a GOP, the reference picture -2 will point to frame -22 ($-2-(6-1)\times 4 = -22$) in the original video sequence. We can conclude that both parallel and sequential algorithms will produce different bit streams. We will analyze, in Sect. 5, the impact of this fact in terms of PSNR and bit rate.

In Fig. 2, we can see a representation of the Option II parallel distribution. As in Option I, all processes, except the root process, encode their first B-frame without a nearby I-frame reference picture. The root process encodes the first I-frame. The improvement respect to the Option I algorithm is that the reference pictures are not significantly disturbed, because each process works with a large number of adjacent GOPs. In the previous example, for the second frame of the GOP, the pattern is only

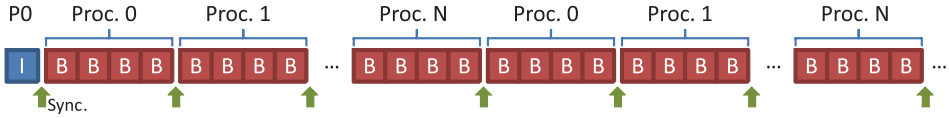


Fig. 1 Option I parallel distribution

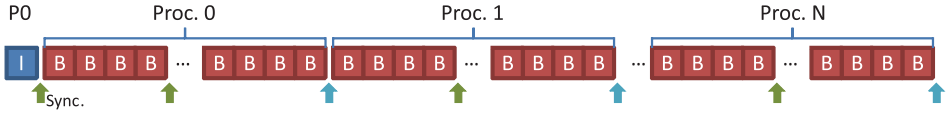


Fig. 2 Option II parallel distribution

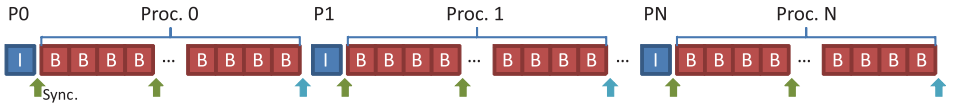


Fig. 3 Option III parallel distribution

Table 1 *IntraPeriod* parameter to match sequential and parallel execution in Option III

Number of processes	240 frames	480 frames
2	120	240
4	60	120
6	40	80
8	30	60
10	24	48
12	20	40

Fig. 4 Option IV parallel distribution

altered for the first three GOPs of a thread. From this point onward, all reference pictures needed are available in the private working buffer of each process. To minimize the distortion of the reference pictures, we assign the maximum number of adjacent GOPs to each process.

Figure 3 shows the parallel distribution of Option III, where the parallel structure is similar to Option II. In Option II, we assign one group of adjacent GOPs per process. In Option III, moreover, we consider each group as a video sequence, and start the encoding procedure of each process computing the first frame as an I-frame. In this case, the parallel Option III and sequential executions can be exactly the same if in the sequential execution we do a slight change in the standard configuration.

To get the same bit stream with both the parallel and sequential algorithms, we must change the *IntraPeriod* parameter according to the number of processes of the parallel execution. Table 1 shows the value of the *IntraPeriod* parameter when we compute 240 and 480 frames. Moreover, the I-frame included must be an IDR (Instantaneous Decoding Refresh), so we set the *DecodingRefreshType* parameter equal to 2.

Finally, in Fig. 4, the parallel distribution for Option IV is shown. Note that the parallel structure is similar to the parallel structure of Option I, but the GOP consists of one I-frame, therefore there are no differences between the parallel and the sequential execution.

5 Numerical experiments

The proposed algorithms have been tested on a shared memory platform evaluating parallel performance, PSNR and bit rate. The multicore platform used consists of two Intel XEON X5660 hexacores at up to 2.8 GHz and 12MB cache per processor, and 48 GB of RAM. The operating system used is CentOS Linux 5.6 for x86 64 bit. The parallel environment has been managed using OpenMP [17]. The compiler used was $g{+}{+}$ compiler $v.4.1.2$.

Table 2 Test video sequences

Acronym	File name	Width	Height	Frame rate	Total frames
TF	Traffic.yuv	2,560	1,600	30	150
BQ	BQTerrace.yuv	1,920	1,080	60	600
FP	FourPeople.yuv	1,280	720	60	600
RH	RaceHorses.yuv	832	480	30	300
BP	BasketballPass.yuv	416	240	50	500

The testing video sequences used in our experiments are listed in Table 2 and we present results for them using LB mode and AI mode, encoding 120, 240 and 480 frames.

In Fig. 5, we present the computation times for Option I, Option II, Option III, and Option IV parallel algorithms over FP, RH and BP test video sequences encoding 240 frames. The results show a good parallel behavior in all cases. On the other hand, all parallel algorithms using just 1 process are identical to the sequential version, therefore the sequential reference algorithm for Options I, II and III is the same sequential algorithm (the LB sequential algorithm), while the reference algorithm for Option IV is the sequential AI execution.

Figure 6 shows the speed-up associated to the results shown in Fig. 5. This figure confirms the good behavior of all proposed parallel algorithms, obtaining nearly ideal efficiencies in some cases. Note that the speed-up increases as the number of processes increases up to the maximum number of available cores, and the ratio of increase remains constant. In Fig. 7, we compare the speed-up obtained by all proposed parallel algorithms when encoding 240 frames of the BQ test video sequence. As it can be seen, Option III obtains the best results for the LB mode being the behavior similar in the rest of tested video sequences. Also, Option IV (AI mode) obtains similar speed-ups to Option III (LB mode).

Figure 8 shows both PSNR and bitrate evolution as a function of the number of cores used for RH video sequence. As it can be seen, Option I and Option II approaches produce a bitrate increase as the number of processes increases. Option I algorithm drastically changes the structure of the reference pictures, and as a consequence it causes the large bit rate increase. Note that the first frames of each process are encoded without nearby reference frames, increasing the bit rate as the number of processes does. Finally, the bit rate increase is greater in Option III because the initial frame is an I-frame, whereas the initial frame in Option II algorithm is a B-frame. In GOP-based parallel algorithms, this fact cannot be avoided without altering the quality criterion. However, in Option III approach, the increased bitrate is lower when the number of total encoded frames is increased and also, the quality improves as the bitrate increases.

Finally, we have modified the low-delay profile configuration to obtain the same PSNR and bit rate results using both parallel and sequential versions of Option III algorithm and have encoded 480 frames of BP video sequence. In Fig. 9, *NON_EQUIV* curve represents the speed-up achieved when parallel and sequential algorithms obtain slightly different results, and *EQUIV* curve represents the speed-up achieved when they provide equivalent executions. We can conclude that the proposed Option III algorithm obtains a good efficiency with slight divergences.

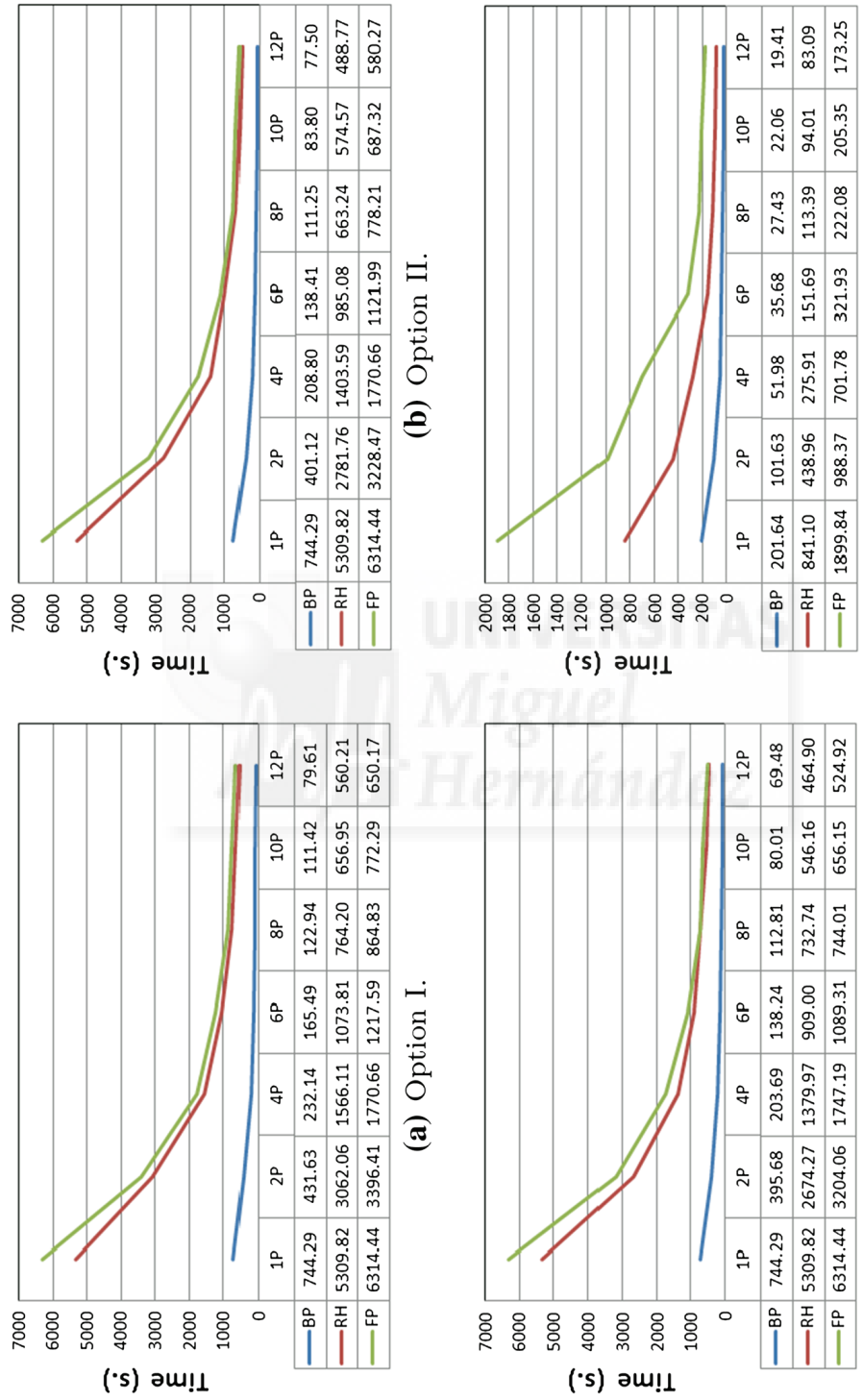


Fig. 5 Computational times for the parallel algorithms (FP, RH and BP). 240 frames

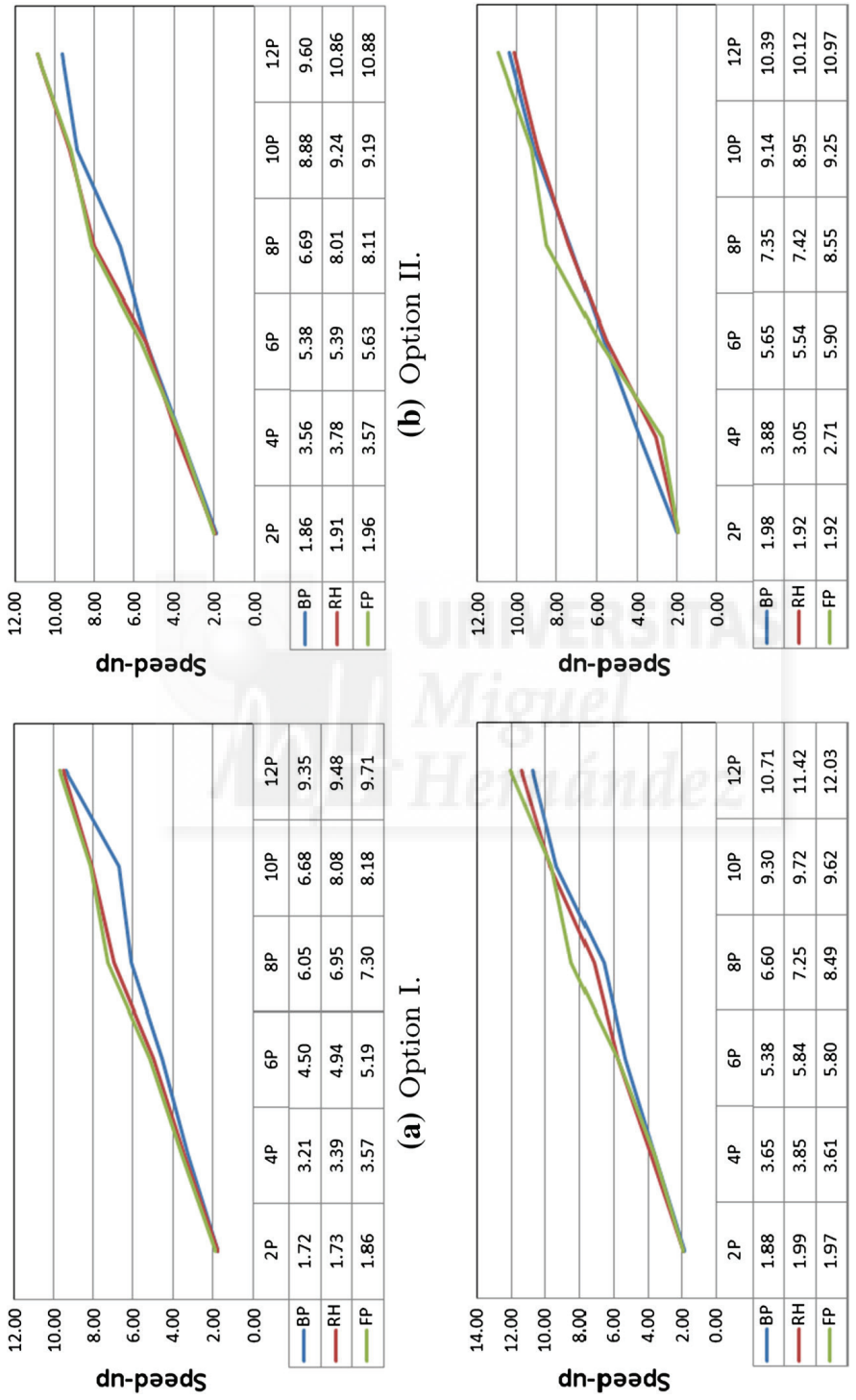
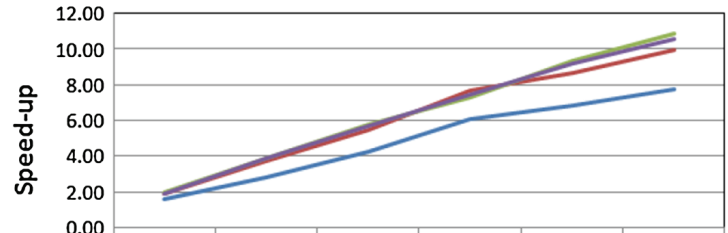


Fig. 6 Speed-up for the parallel algorithms (FP, RH and BP). 240 frames



	2P	4P	6P	8P	10P	12P
Opt. I	1.60	2.79	4.31	6.08	6.89	7.78
Opt. II	1.93	3.72	5.47	7.69	8.71	9.98
Opt. III	1.97	3.86	5.76	7.26	9.33	10.87
Opt. IV	1.93	3.95	5.74	7.48	9.24	10.61

Fig. 7 Speed-up for the parallel algorithm BQ. 240 frames

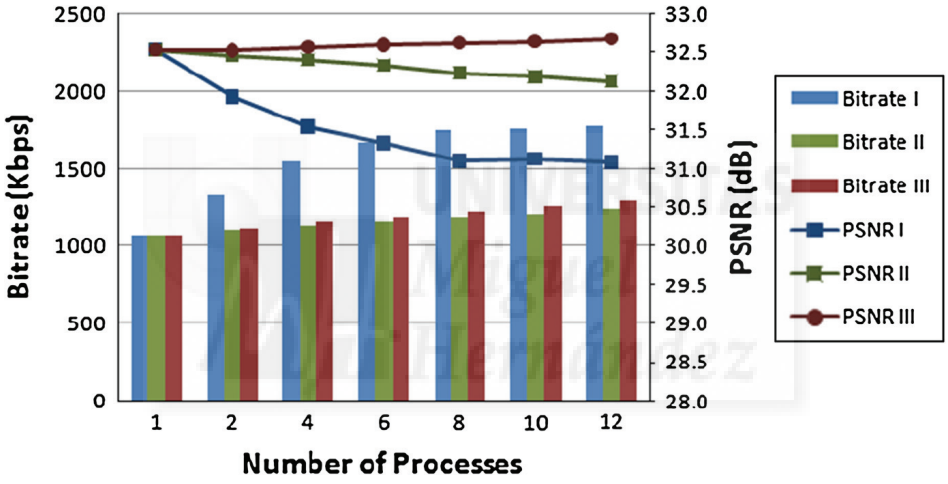
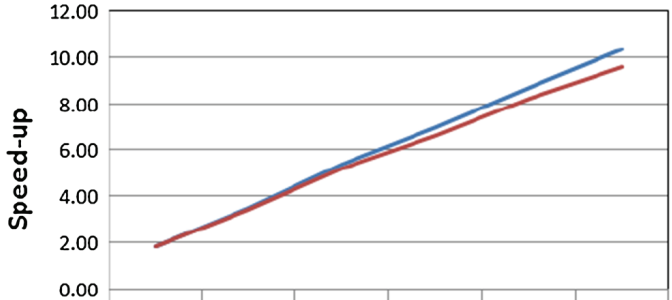


Fig. 8 Rate-distortion for RH sequence, encoding 240 frames with different number of processes



	2 Proc.	4 Proc.	6 Proc.	8 Proc.	10 Proc.	12 Proc.
NON_EQUIV	1.84	3.51	5.40	6.97	8.73	10.36
EQUIV	1.82	3.44	5.19	6.65	8.22	9.60

Fig. 9 Speed-up for Option III parallel algorithms. BP video sequence encoding 480 frames

As we have said, the algorithm proposed in [11], achieves a speed-up of up to 5× when using 7 cores. Both the cited algorithm and our Option IV approaches, encode a video sequence using intra mode. Note that Figs. 6 and 7 show results of a speed-up above 5.5× using 6 processes (or cores). As shown, our parallel algorithm obtains better speed-ups using less processes.

6 Conclusions

In this paper, we have proposed several parallel algorithms for HEVC video encoder. These algorithms are based on a coarser grain parallelization approach with the organization of video frames in GOPs and different GOP process allocation schemes. They are specially suited for multicore architectures. After implementing the algorithms in HEVC software, some experiments were performed showing interesting results as follows (a) GOP organization determines the final performance in terms of speed-up/efficiency, being the best approach Option IV (AI mode) when comparing both sequential and parallel versions, and (b) both Option II and Option III algorithms introduce a bit rate overhead as the number of processes increases being greater in Option III, and in Option III algorithm the PSNR slightly increases. In general, all proposed versions attain high parallel efficiency results, showing that GOP-based parallelization approaches should be taken into account to reduce the HEVC video encoding complexity. As future work, we will explore hierarchical parallelization approaches combining GOP-based approaches with slice and wavefront parallelization levels.

References

1. ITU-T and ISO/IEC JTC 1 (2012) Advanced video coding for generic audiovisual services, ITU-T Rec. H.264 and ISO/IEC 14496–10 (AVC) version 16
2. Ohm J, Sullivan G, Schwarz H, Tan TK, Wiegand T (2012) Comparison of the coding efficiency of video coding standards - including high efficiency video coding (hevc). IEEE Trans Circuits Systems Video Technol 22(12):1669–1684
3. Bross B, Han W, Ohm J, Sullivan G, Wang Y-K, Wiegand T (2013) High efficiency video coding (HEVC) text specification draft 10, Document JCTVC-L1003 of JCT-VC, Geneva, January 2013
4. Sullivan G, Ohm J, Han W, Wiegand T (December 2012) Overview of the high efficiency video coding (HEVC) standard. IEEE Trans Circuits Systems Video Technol 22(12):1648–1667
5. Bossen F, Bross B, Suhring K, Flynn D (2012) HEVC complexity and implementation analysis. IEEE Trans Circuits Systems Video Technol 22(12):1685–1696
6. Alvarez-Mesa M, Chi C, Juurlink B, George V, Schierl T (2012) Parallel video decoding in the emerging HEVC standard. In: International conference on acoustics, speech, and signal processing, Kyoto, March 2012, pp 1–17
7. Ayele E, Dhok SB (2012) Review of proposed high efficiency video coding (HEVC) standard. Int J Comput Appl 59(15):1–9
8. Chi C, Alvarez-Mesa M, Lucas J, Juurlink B, Schierl T (2013) Parallel hevc decoding on multi- and many-core architectures. J Signal Process Systems 71(3):247–260. Available http://dx.doi.org/10.1007/s11265-012-0714-2
9. Yu Q, Zhao L, Ma S (2012) Parallel AMVP candidate list construction for HEVC. In: VCIP'12, 2012, pp 1–6
10. Jiang J, Guo B, Mo W, Fan K (2012) Block-based parallel intra prediction scheme for HEVC. J Multimed 7(4):289–294
11. Zhao Y, Song L, Wang X, Chen M, Wang J (2013) Efficient realization of parallel HEVC intra encoding. In: IEEE international conference on multimedia and expo workshops (ICMEW), July 2013, pp 1–6

12. x265 project. http://code.google.com/p/x265
13. Bossen F (2013) Common test conditions and software reference configurations. Joint Collaborative Team on Video Coding, Geneva, Tech. Rep. JCTVC-L1100, January 2013
14. Bjontegaard G (2008) Improvements of the BD-PSNR model. Video Coding Experts Group (VCEG), Berlin (Germany), Tech. Rep. VCEG-M33, July 2008
15. HEVC Reference Software. https://hevc.hhi.fraunhofer.de/svn/svnHEVCSoftware/tags/HM-10.0/
16. Marpe D, Schwarz H, Wiegand T (2003) Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard. Circuits Systems Video Technol IEEE Trans 13(7):620–636
17. Openmp application program interface, version 3.1. OpenMP Architecture Review Board. http://www.openmp.org, 2011

J Supercomput (2015) 71:1882–1892
DOI 10.1007/s11227-014-1371-y

Slice-based parallel approach for HEVC encoder

P. Piñol · H. Migallón · O. López-Granado · M. P. Malumbres

Published online: 30 December 2014

Abstract The high efficiency video coding (HEVC) is the newest video coding standard from the ITU-T Video Coding Experts Group and the ISO/IEC Moving Picture Experts Group, which significantly increases the computing demands to encode video to reach the limits on compression efficiency. Our interest is centered on applying parallel processing techniques to HEVC encoder to significantly reduce the computational time without disturbing the coding performance behavior. We propose a parallelization approach to the HEVC encoder which is well suited to multicore architectures. Our proposal uses OpenMP programming paradigm working at slice parallelization level. We encode several slices of each frame at the same time using all available processing cores. The results show that speed-ups up to 9.8 can be obtained for the All Intra mode and up to 8.7 for Low-Delay B, Low-Delay P and Random Access modes for 12 processes with a negligible loss in coding performance.

Keywords Parallel algorithms · Video coding · HEVC · Multicore · Performance

1 Introduction

The new high efficiency video coding (HEVC) standard has been recently developed by the Joint Collaborative Team on video coding (JCT-VC) which was established by the ISO/IEC Moving Picture Experts Group (MPEG) and ITU-T Video Coding

This research was supported by the Spanish Ministry of Education and Science under grant TIN2011-27543-C03-03, the Spanish Ministry of Science and Innovation under grant TIN2011-26254.

P. Piñol · H. Migallón (✉) · O. López-Granado · M. P. Malumbres
Physics and Computer Architecture Department,
Miguel Hernández University, 03202 Elche, Spain
e-mail: hmigallon@umh.es

Experts Group (VCEG). This new standard will replace the current H.264/AVC [1] standard to deal with nowadays and future multimedia market trends. 4K definition video content is a nowadays fact and 8K is the next step. The HEVC standard aims to improve coding efficiency with respect to the H.264/AVC High profile, delivering the same video quality at half the bit-rate [2].

Regarding complexity, HEVC encoder is several times more complex than H.264/AVC encoder, so reducing its complexity will be a hot research topic in the years to come. At the time of developing this work, the current version of the reference software, called HEVC test model (HM), is HM 10.0 [3] which corresponds to the HEVC text specification draft 10 [4]. A good overview of HEVC standard can be found in [5].

We can find in the literature several works about complexity analysis and parallelization strategies for the emerging HEVC standard as in [6–8].

Most of the publications about HEVC parallelization, are focused on the decoding side, while a small number of publications are focused on the encoding side. Looking for the most appropriate parallel optimizations at the decoder that provide real-time decoding of high-definition (HD) and ultra-high-definition (UHD) video contents [7]. In [9] and [10] the authors present a variation of wavefront parallel processing (WPP) called overlapped wavefront (OWF) for the HEVC decoder in which the executions over consecutive pictures are overlapped. When a thread finishes the processing of a coding tree block (CTB) row, it can begin processing the next picture instead of waiting until the end of the current picture. Recently, in [11] the authors mixed tiles, WPP and SIMD instructions to develop a real-time HEVC decoder.

Currently, there are few works focused on the HEVC encoder. In [12] the authors propose a fine-grain parallel optimization in the motion estimation module of the HEVC encoder, allowing to perform the motion vector prediction in all prediction units (PUs) available at the coding unit (CU) at the same time. In [13] the authors propose a parallelization inside the intra-prediction module that consists on removing data dependencies between sub-blocks of a CU, obtaining interesting speed-up results. Other recent works are focused on changes in the scanning order. For example, in [14] the authors propose a CU scanning order based on a diamond search obtaining a good scheme for massive parallel processing. Also in [15] the authors propose to change the HEVC deblocking filter processing order obtaining time savings of 37.93 % over many-core processing.

In this paper we present a parallelization for the HEVC encoder at slice level. In this approach, the number of slices per frame varies as a function of the number of processes used to encode the sequence. Depending on the number of processes used, each slice will contain a different number of consecutive CUs. Furthermore, we analyze the overall behavior in terms of complexity reduction and coding performance.

The remainder of this paper is organized as follows: Sect. 2 presents a brief overview of HEVC and slices. Section 3 shows the proposed slice-based parallelism strategy. In Sect. 4 we analyze the performance of the proposed parallel algorithm. Finally, in Sect. 5 some conclusions are drawn.

2 HEVC and slices

HEVC is a video codec that uses a block-based hybrid video coding scheme. This scheme is based in motion estimation/compensation and transform coding. In the encoding process, each frame is divided into blocks called coding units (CU). The maximum size of a CU in HEVC is 64×64 pixels. A CU can be divided into smaller blocks for the encoding process. These blocks can be encoded using one of three modes: (a) without any prediction, (b) with spatial prediction, or (c) with temporal prediction. Spatial prediction exploits spatial redundancy within a frame. To encode a block, it uses previous regions of the same frame to calculate a block candidate and then encodes only the error of that estimation. Temporal prediction uses previously encoded frames to estimate the block candidate, by means of a search of a similar block in other frames (called reference frames). It exploits temporal redundancy, taking advantage of the fact that nearby frames usually contain blocks which are very similar to the current block and so the residuum of the compensation is close to zero. For the three encoding modes a discrete cosine transform (DCT) is applied to the resulting values so they are converted into the frequency domain. Then, these coefficients are quantized and the result is compressed by an entropy encoder.

A block is referred to as "intra" when no information of other frames is used to encode it [modes (a) and (b)]. A block is referred to as "inter" when temporal prediction is used [mode (c)].

Slices are partitions of an encoded frame which can be independently decoded (regarding the other slices of the same frame). As a slice can be decoded without any information from other slices, neither intra-prediction nor inter-prediction can cross slice boundaries using CUs from other slices of the same frame. So, every single slice can be encoded/decoded in an isolated way. Slices consist of an integer number of CUs. An I-slice is composed of intra CUs. P-slices and B-slices can contain both intra and inter CUs. Here 'P' stands for unidirectional prediction and 'B' stands for bidirectional prediction. CUs in a P-slice can only use one reference frame. CUs in a B-slice can use up to two reference frames, i.e., the estimation and compensation mechanism can use the combination of up to two blocks in different reference frames. One frame can be composed of only one slice or several slices.

In the configuration files provided within reference software package [3], four coding setups can be found: All Intra, Random Access, Low-Delay B, and Low-Delay P.

In All Intra (AI) setup every frame is coded as an I-frame (all its slices are I-slices), i.e., it is coded without any motion estimation/-compensation. So each frame is independent from the other frames in the sequence. This mode gets lower compression rates (compared to the other 3 setups) because P-frames and B-frames can usually obtain better compression rates than I-frames at the same quality level. On the other hand, the encoding process for AI mode is faster than for the other three setups because no time is wasted in motion estimation. Every frame is coded in rendering order. Applications that require a fast encoding process and are not concerned about limited bandwidth or storage capacity, fit perfectly in this coding setup.

Random Access (RA) setup combines I-frames and B-frames in Group Of Pictures (GOPs) of eight frames. B-frames usually achieve good compression rates. Each block

of a B-frame can use up to two reference frames, so in the encoding process two lists of reference pictures are maintained. Reference frames are located earlier and later than the frame we are currently coding. In this setup, coding (and decoding) order is not the same as rendering order. To allow navigating along the coded sequence (pointing to a certain moment) or to allow functions like fast forward, an I-frame is inserted periodically. Depending on the frame rate of each sequence the intra refresh period varies. The intra period is a multiple of 8 (the size of the GOP) which inserts an I-frame approximately every second. Applications that do not have time constraints (when coding a video sequence) and need features like the aforementioned fast forward, are the target applications of this coding mode.

Low-Delay setups (LP and LB) encode each frame in rendering order. First an I-frame is inserted in the coded bit stream and then only P-frames (or B-frames) are used for the rest of the video sequence. As stated before, a B-frame uses a pair of frames to perform motion estimation and compensation. In LB mode this pair of frames is always selected from previous frames (in rendering order). On the contrary, RA mode uses past and future frames as reference for B-frames and this introduces a delay because you have to wait for future pictures to be available to encode/decode the present frame. In LP and LB setups GOP size is equal to 4. These two modes achieve better compression performance than AI mode and do not suffer from the delay introduced by RA mode. Applications like video-conference which have bandwidth and time constraints can benefit from low-delay modes.

3 Parallel algorithms

In our parallel proposal of HEVC encoder we have divided each video frame in as many slices as the available number of parallel processes, in such a way that if we are using 12 parallel processes we will divide each frame into 12 slices and each slice will be encoded by a different process. In RA, LP and LB setups, a frame used as a reference picture has to be available as a whole for every future slice that uses it for motion estimation/compensation. A synchronization mechanism is imposed by this restriction. Every process encodes a different slice in a parallel way but all those processes need to be synchronized before encoding the next frame. In AI mode this is not strictly necessary as there are no reference pictures but we have followed the same synchronizing scheme for AI setup. In RA, LP and LB setups all decoded slices of the reference frames need to be available to encode the following frame, so to avoid unnecessary sending of slices or decoded pieces of a frame between processes, we have used a shared memory approximation to store the decoded picture buffer.

In our tests we have used three video sequences named “Four People (FP)” (which has a resolution of 1280×720 pixels), “Race Horses (RH)” (with a resolution of 832×480 pixels) and “BQ Terrace (BQ)” (with a resolution of 1920×1280 pixels). We have performed different tests by dividing frames into 1, 2, 4, 6, 8, 10 and 12 slices and using the corresponding number of parallel processes. Note that in our experiments we obtain slices which have a similar number of CUs each, but this is absolutely not a restriction. We have chosen a uniform partition of the frame to homogenize the time that each process spends in encoding its slice so as to reduce the dead time

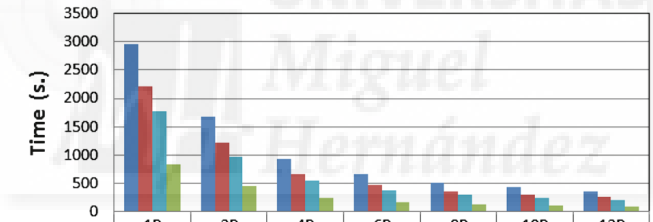
of the processes (caused by the synchronization mechanism). Nevertheless we have observed that the time that each process needs to encode its slice can diverge even if slices have all the same size. The reason is that algorithms like motion estimation or entropy encoding can differ in processing time for different regions of a frame.

4 Numerical experiments

The proposed algorithms have been tested on a shared memory platform evaluating parallel performance, PSNR and bit-rate. The multicore platform used consists of two Intel XEON X5660 hexacores at up to 2.8 GHz and 12 MB cache per processor, and 48 GB of RAM, with no hyperthreading enabled. The operating system used is CentOS Linux 5.6 for $\times$86 64 bit. The parallel environment has been managed using OpenMP [16]. The compiler used was g++ compiler v.4.1.2.

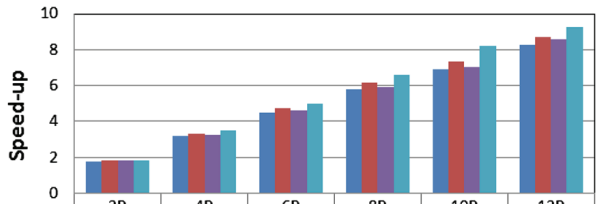
As we have said the testing video sequences used in our experiments are RH, FP and BQ, and we present results using LB, LP, RA and AI modes, encoding 120 frames at different quantization parameters (QPs) (22, 27, 32, 37).

In Fig. 1 we present both the computational times and the speed-ups for the FP video sequence using AI, LB, LB and RA modes, processing 120 frames. As it can be seen the proposed parallel algorithm obtains speed-ups of up to 9.3 for 12 cores in AI



	1P	2P	4P	6P	8P	10P	12P
LowDelay-B	2964.7	1672.9	937.1	661.9	512.3	430.3	358.1
Random	2219.8	1215.4	668.7	467.8	361.0	302.5	255.8
LowDelay-P	1781.0	972.6	545.9	388.0	301.0	252.8	207.1
Intra	842.4	458.7	242.9	169.5	128.2	102.7	91.3

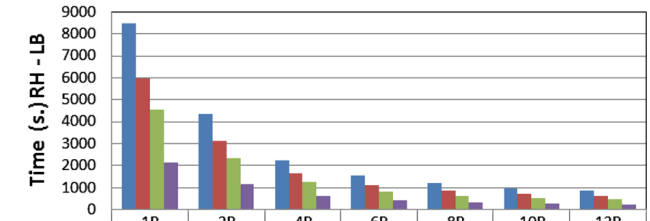
(a) Computational times.



	2P	4P	6P	8P	10P	12P
LowDelay-B	1.8	3.2	4.5	5.8	6.9	8.3
Random	1.8	3.3	4.7	6.1	7.3	8.7
LowDelay-P	1.8	3.3	4.6	5.9	7.0	8.6
Intra	1.8	3.5	5.0	6.6	8.2	9.3

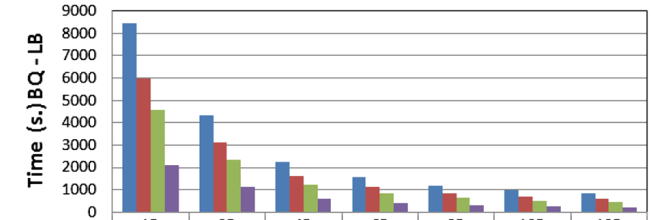
(b) Speed-ups.

Fig. 1 Computational times and speed-ups for the FP video sequence encoded in AI, LB, LP and RA modes processing 120 frames with different number of processes at QP = 32



	1P	2P	4P	6P	8P	10P	12P
LowDelay-B	8467.8	4341.9	2259.2	1553.2	1184.3	971.6	836.5
Random	5958.1	3142.4	1627.3	1108.2	843.1	690.4	603.0
LowDelay-P	4555.1	2351.7	1229.3	832.2	634.7	515.6	456.2
Intra	2114.7	1135.8	612.9	423.0	304.2	250.4	218.1

(a) RH video sequence.



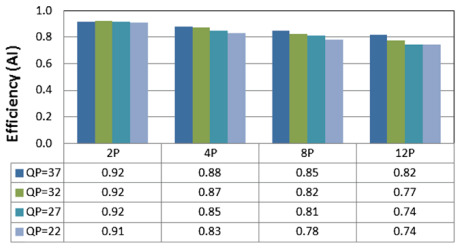
	1P	2P	4P	6P	8P	10P	12P
LowDelay-B	8467.8	4341.9	2259.2	1553.2	1184.3	971.6	836.5
Random	5958.1	3142.4	1627.3	1108.2	843.1	690.4	603.0
LowDelay-P	4555.1	2351.7	1229.3	832.2	634.7	515.6	456.2
Intra	2114.7	1135.8	612.9	423.0	304.2	250.4	218.1

(b) BQ video sequence.

Fig. 2 Computational times for RH and BQ video sequences 120 frames with different number of processes at QP = 32

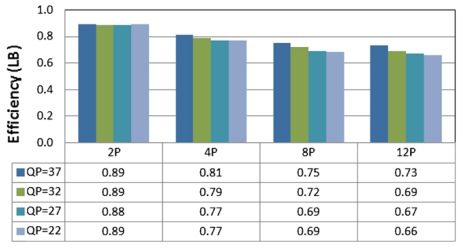
mode and efficiencies of 0.8 on average. We can observe that AI mode obtains better efficiencies than LB, LP and RA modes because motion estimation/compensation process applied in LB, LP and RA modes produces different residual data on each slice and, therefore, the time required by the entropy encoder to process that residual data in each slice may diverge from one slice to another. Figure 2 shows computational times for RH and BQ video sequences setting QP equal to 32. As expected, Figs. 1 and 2 confirm that the computational behavior is similar for the three different video sequences evaluated. We can extend this behavior for the rest of QP values.

In Fig. 3 we present a comparison of efficiencies as a function of the number of processes as well as the compression ratio. As it can be seen, good efficiencies are obtained for all modes. Note that, there is an efficiency loss as the number of cores increases because of the different time required to encode each slice, even so, the parallel algorithm offers good scalability, i.e., the efficiency loss is low. That efficiency loss is lower at high compression ratios, where residual data tend to be more similar in each slice after a higher quantization and so, a similar time is required by each process to encode that residual data. Therefore, the time spent by threads waiting for the slowest one is reduced. On average, efficiencies of 0.75, 0.78, 0.79 and 0.83 are obtained for LB, LP, RA and AI mode, respectively. Figure 4 shows efficiencies when RH and BQ video sequences are encoded setting QP equal to 32, obtaining similar



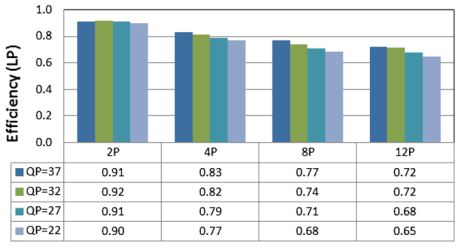
	2P	4P	8P	12P
QP=37	0.92	0.88	0.85	0.82
QP=32	0.92	0.87	0.82	0.77
QP=27	0.92	0.85	0.81	0.74
QP=22	0.91	0.83	0.78	0.74

(a) All Intra mode.



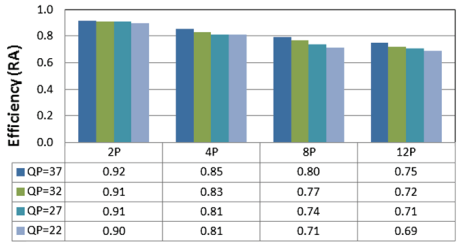
	2P	4P	8P	12P
QP=37	0.89	0.81	0.75	0.73
QP=32	0.89	0.79	0.72	0.69
QP=27	0.88	0.77	0.69	0.67
QP=22	0.89	0.77	0.69	0.66

(b) Low-Delay B mode.



	2P	4P	8P	12P
QP=37	0.91	0.83	0.77	0.72
QP=32	0.92	0.82	0.74	0.72
QP=27	0.91	0.79	0.71	0.68
QP=22	0.90	0.77	0.68	0.65

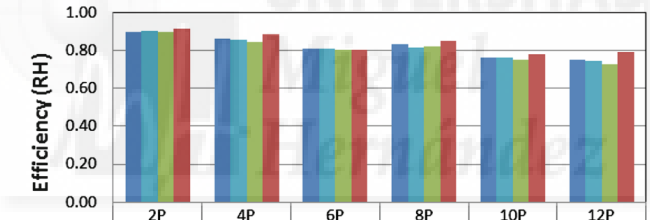
(c) Low-Delay P mode.



	2P	4P	8P	12P
QP=37	0.92	0.85	0.80	0.75
QP=32	0.91	0.83	0.77	0.72
QP=27	0.91	0.81	0.74	0.71
QP=22	0.90	0.81	0.71	0.69

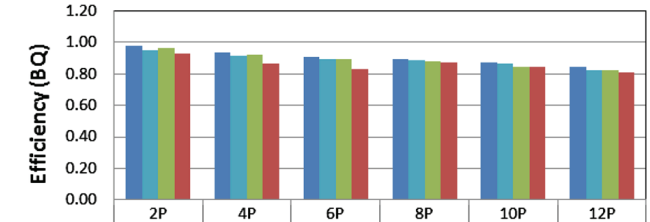
(d) Random Access mode.

Fig. 3 Efficiencies for the FP video sequence with different number of processes and QPs, processing 120 frames



	2P	4P	6P	8P	10P	12P
LowDelay-B	0.90	0.86	0.81	0.83	0.76	0.75
Random	0.91	0.86	0.81	0.82	0.76	0.75
LowDelay-P	0.90	0.85	0.80	0.82	0.75	0.73
Intra	0.92	0.89	0.80	0.85	0.78	0.79

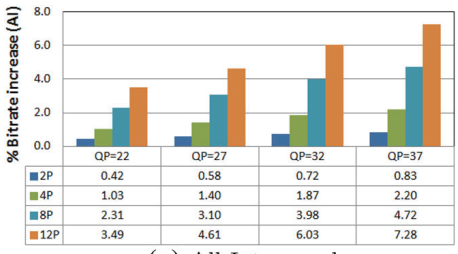
(a) RH video sequence.



	2P	4P	6P	8P	10P	12P
LowDelay-B	0.98	0.94	0.91	0.89	0.87	0.84
Random	0.95	0.92	0.90	0.88	0.86	0.82
LowDelay-P	0.96	0.92	0.89	0.88	0.84	0.82
Intra	0.93	0.86	0.83	0.87	0.84	0.81

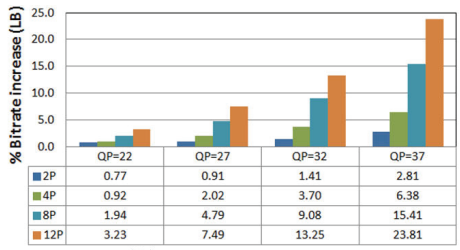
(b) BQ video sequence.

Fig. 4 Efficiencies for the RH and BQ video sequences at QP = 32, processing 120 frames



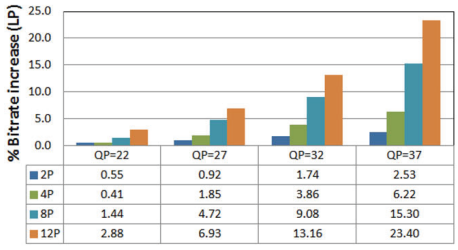
	QP=22	QP=27	QP=32	QP=37
2P	0.42	0.58	0.72	0.83
4P	1.03	1.40	1.87	2.20
8P	2.31	3.10	3.98	4.72
12P	3.49	4.61	6.03	7.28

(a) All Intra mode.



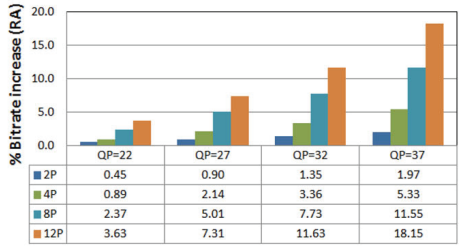
	QP=22	QP=27	QP=32	QP=37
2P	0.77	0.91	1.41	2.81
4P	0.92	2.02	3.70	6.38
8P	1.94	4.79	9.08	15.41
12P	3.23	7.49	13.25	23.81

(b) Low-Delay B mode.



	QP=22	QP=27	QP=32	QP=37
2P	0.55	0.92	1.74	2.53
4P	0.41	1.85	3.86	6.22
8P	1.44	4.72	9.08	15.30
12P	2.88	6.93	13.16	23.40

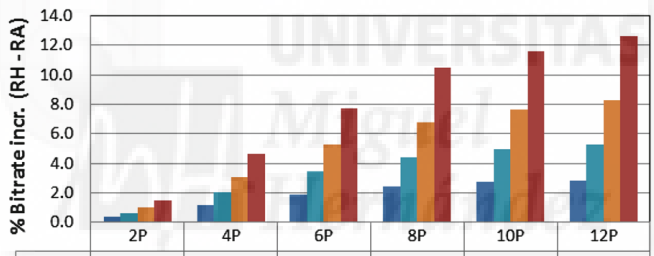
(c) Low-Delay P mode.



	QP=22	QP=27	QP=32	QP=37
2P	0.45	0.90	1.35	1.97
4P	0.89	2.14	3.36	5.33
8P	2.37	5.01	7.73	11.55
12P	3.63	7.31	11.63	18.15

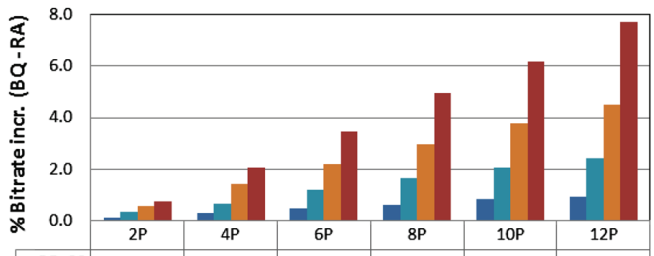
(d) Random Access mode.

Fig. 5 Evolution of bit-rate increment (%) for FP video sequence with different number of processes as a function of the QP parameter, encoding 120 frames



	2P	4P	6P	8P	10P	12P
QP=22	0.37	1.20	1.88	2.42	2.73	2.83
QP=27	0.62	2.07	3.44	4.41	4.95	5.30
QP=32	0.99	3.07	5.28	6.80	7.62	8.24
QP=37	1.51	4.68	7.69	10.47	11.62	12.66

(a) RH video sequence.



	2P	4P	6P	8P	10P	12P
QP=22	0.12	0.32	0.49	0.61	0.85	0.95
QP=27	0.35	0.66	1.21	1.65	2.06	2.43
QP=32	0.58	1.45	2.20	2.96	3.77	4.50
QP=37	0.74	2.07	3.45	4.97	6.18	7.71

(b) BQ video sequence.

Fig. 6 Evolution of bit-rate increment (%) for RH and BQ video sequences encoded in RA mode with different number of processes as a function of the QP parameter encoding 120 frames

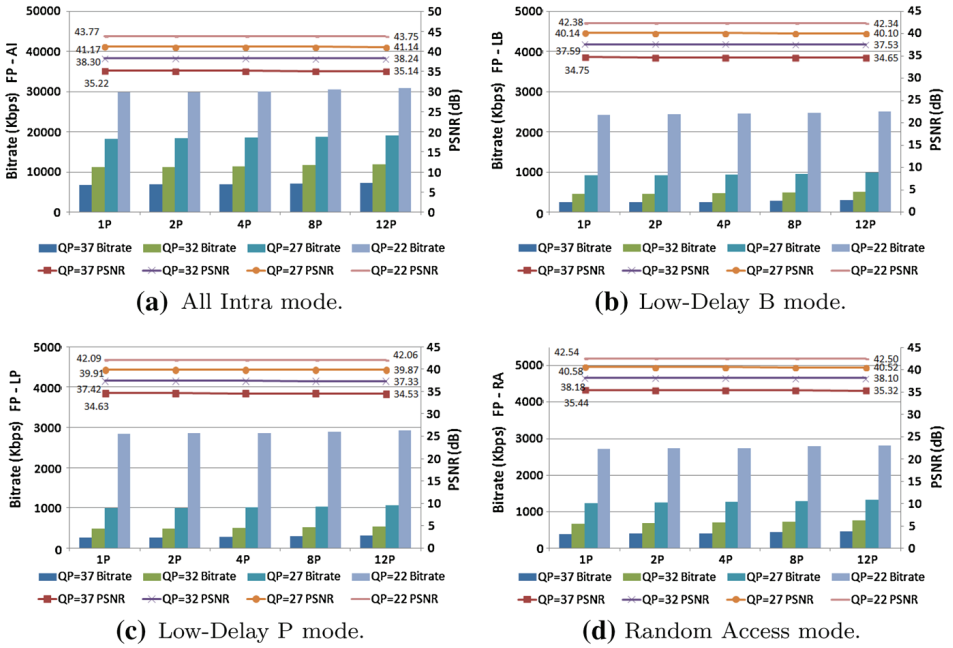


Fig. 7 R/D for FP video sequence with different number of processes, encoding 120 frames

results for the FP video sequence. Also it can be noticed the results improve for a high resolution video sequence (BQ).

In Fig. 5 we show the bit-rate overhead as a function of the number of processes and the quantization parameter (QP). As it can be seen, all modes increase the bit-rate as the number of slices and the compression ratio increase. In AI mode that overhead appears due to the high number of headers used and the reduced number of CUs available for the intra-prediction process at each slice. As shown in Fig. 5a, the maximum bit-rate increment for AI mode is 7.28 % with 12 slices per frame and a QP value of 37, being on average lower that 3 %. In the rest of modes there is a similar behavior increasing the bit-rate as the number of slices and quantization parameter do. The bit-rate overhead in LB, LP and RA modes is also due to the high number of headers and because motion vectors cannot be predicted across slice boundaries. On the other hand CABAC (the HEVC entropy encoder) context models are not updated after the computation of each slice but after the frame computation. As shown in Fig. 5, in LB, LP and RA modes that overhead has a greater impact in the final bitstream, being the maximum bit-rate increment equal to 23.81, 23.40 and 18.15 % for LB, LP and RA modes, respectively. Figure 6 presents the results for RH and BQ video sequences encoded in RA mode. These results show that the bit-rate overhead is even lower in RH and BQ video sequences, with respect to FP video sequence.

Regarding video quality, in Fig. 7 we present rate/distortion (R/D) information for all modes. These figures represent the bit-rate (vertical bars) and PSNR (horizontal lines) produced by the parallel algorithm using different number of cores (up to 12) and compressing at different QPs. As it can be seen, in all cases there are slight differences in both PSNR and bit-rate between the sequential algorithm (with one slice per frame)

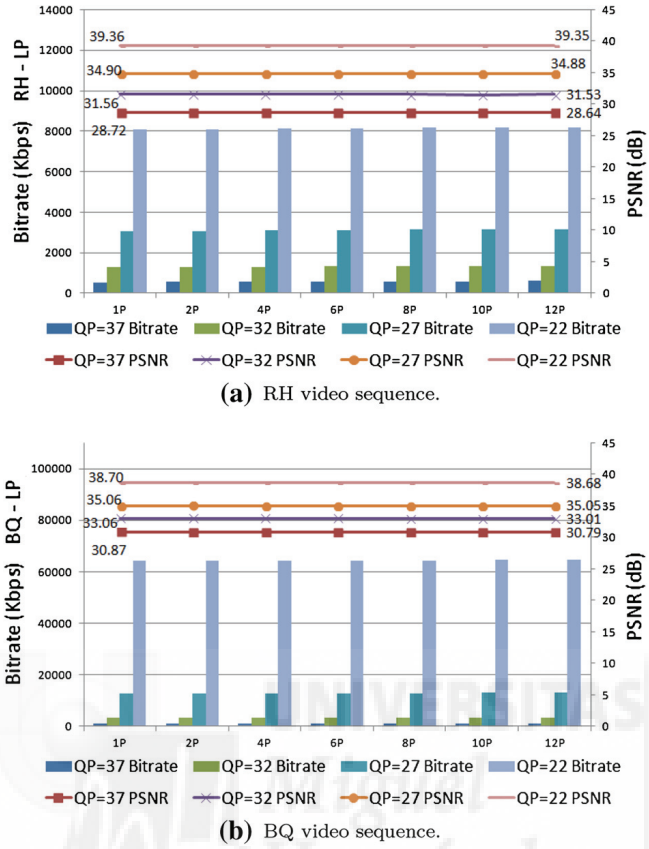


Fig. 8 R/D for RH and BQ video sequences, using LP mode with different number of processes and encoding 120 frames

and the parallel version (with two or more slices per frame), and keeping the increment of bit-rate lower than 23.8 % at high compression ratios (QP = 37) using 12 processes when encoding 120 frames. As previously said, as the number of slices increases, a greater number of headers are included in the final bitstream and this has a greater impact at high compression ratios. Figure 8 shows R/D results for RH and BQ video sequences using LP mode. Note that the R/D behavior remains the same.

5 Conclusions

In this paper we have proposed a parallel algorithm for the HEVC video encoder. This algorithm is based on a slice parallelization approach, dividing a frame into different number of slices depending on the number of processes used. After implementing the algorithms in HEVC software, some experiments were performed for All Intra, Low-Delay B, Low-Delay P and Random Access modes. The results show that speed-ups up to 9.8 can be obtained for the AI mode and up to 8.8 for LB, LP and RA mode at QP = 37 when encoding 120 frames using 12 processes. Regarding coding performance, there

is an increase in the bit-rate in both coding modes, being on average lower than 6 % for LB, LP and RA modes and being on average lower than 3 % for the All Intra mode. The quality loss is negligible in all experiments reported. In general, the proposed version attains good parallel efficiency results, showing that parallelization approaches should be taken into account to reduce the HEVC video encoding complexity.

References

1. ITU-T and ISO/IEC JTC 1 (2012) Advanced video coding for generic audiovisual services. In: ITU-T recommendation H.264 and ISO/IEC 14496–10 (AVC) version 16
2. Ohm J, Sullivan G, Schwarz H, Tan TK, Wiegand T (2012) Comparison of the coding efficiency of video coding standards—including high efficiency video coding (HEVC). IEEE Trans Circuits Syst Video Technol 22(12):1669–1684
3. HEVC Reference Software (2014) https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-10.0/
4. Bross B, Han W, Ohm J, Sullivan G, Wang Y-K, Wiegand T (2013) High efficiency video coding (HEVC) text specification draft 10. In: Document JCTVC-L1003 of JCT-VC, Geneva
5. Sullivan G, Ohm J, Han W, Wiegand T (2012) Overview of the high efficiency video coding (HEVC) standard. IEEE Trans Circuits Syst Video Technol 22(12):1648–1667
6. Bossen F, Bross B, Suhring K, Flynn D (2012) HEVC complexity and implementation analysis. IEEE Trans Circuits Syst Video Technol 22(12):1685–1696
7. Alvarez-Mesa M, Chi C, Juurlink B, George V, Schierl T (2012) Parallel video decoding in the emerging HEVC standard. In: International conference on acoustics, speech, and signal processing, pp 1–17, Kyoto
8. Ayele E, Dhok SB (2012) Review of proposed high efficiency video coding (HEVC) standard. Int J Comput Appl 59(15):1–9
9. Chi CC, Alvarez-Mesa M, Juurlink B, Clare G, Henry F, Pateux S, Schierl T (2012) Parallel scalability and efficiency of HEVC parallelization approaches. IEEE Trans Circuits Syst Video Technol 22(12):1827–1838
10. Chi C, Alvarez-Mesa M, Lucas J, Juurlink B, Schierl T (2013) Parallel HEVC decoding on multi- and many-core architectures. J Signal Process Syst 71(3):247–260
11. Bross B, Alvarez-Mesa M, George V, Chi CC, Mayer T, Juurlink B, Schierl T (2013) HEVC real-time decoding, pp 88 561R–88 561R–11
12. Yu Q, Zhao L, Ma S (2012) Parallel AMVP candidate list construction for HEVC. In: VCIP'12, pp 1–6
13. Jiang J, Guo B, Mo W, Fan K (2012) Block-based parallel intra prediction scheme for HEVC. J Multimed 7(4):289–294
14. Luczak A, Karwowski D, Mackowiak S, Grajek T (2012) Diamond scanning order of image blocks for massively parallel HEVC compression. In: Bolc L, Tadeusiewicz R, Chmielewski L, Wojciechowski K (eds) Computer vision and graphics, series. Lecture notes in computer science, vol 7594. Springer, Berlin, pp 172–179
15. Yan C, Zhang Y, Dai F, Li L (2013) Efficient parallel framework for HEVC deblocking filter on many-core platform. In: Data compression conference (DCC), pp 530–530
16. OpenMP Architecture Review Board (2011) OpenMP application program interface, version 3.1. OpenMP Architecture Review Board. http://www.openmp.org

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 619379, 9 pages
http://dx.doi.org/10.1155/2014/619379

Research Article

Protection of HEVC Video Delivery in Vehicular Networks with RaptorQ Codes

Pablo Piñol, Miguel Martínez-Rach, Otoniel López, and Manuel Pérez Malumbres

Physics and Computer Architecture Department, Miguel Hernández University, Avenida de la Universidad, s/n, 03202 Elche, Spain

Correspondence should be addressed to Pablo Piñol; pablop@umh.es

Received 11 April 2014; Accepted 25 June 2014; Published 17 July 2014

Academic Editor: Yuxin Mao

With future vehicles equipped with processing capability, storage, and communications, vehicular networks will become a reality. A vast number of applications will arise that will make use of this connectivity. Some of them will be based on video streaming. In this paper we focus on HEVC video coding standard streaming in vehicular networks and how it deals with packet losses with the aid of RaptorQ, a Forward Error Correction scheme. As vehicular networks are packet loss prone networks, protection mechanisms are necessary if we want to guarantee a minimum level of quality of experience to the final user. We have run simulations to evaluate which configurations fit better in this type of scenarios.

1. Introduction

Our lives have experimented a radical change since cell phones are no longer just telephones and have become smartphones, with good processing capabilities, large store capacity, and above all, great connectivity. This connectivity allows us to have all kinds of information available at every moment. And we are not only information consumers but active producers as well. At the day when vehicular networks will be integrated by a high percentage of our vehicles and infrastructures in our cities and roads, a vast number of applications of all types (some of them nowadays unthinkable) will arise. Many of them will be oriented to safety and others to entertainment, economizing, and so forth. Within vehicular networks applications, video streaming can be very useful. But, on the one hand, vehicular networks are inhospitable environments where packet losses appear, and on the other hand, video transmission is heavily resource demanding. The combination of these two characteristics makes video streaming over vehicular networks a hard to manage task.

The aim of this work is to evaluate the protection of video encoded with the emerging standard High Efficiency Video Coding (HEVC) by using RaptorQ codes and how they both behave in vehicular scenarios. For this task we will vary a series of parameters in both HEVC and RaptorQ and will present the performance of those configurations.

There are lots of works that evaluate HEVC efficiency (like [1, 2]) although most of them do not take into consideration lossy environments. Some works evaluate HEVC performance under packet loss conditions as the authors do in [3]. In that work the authors have developed a complete framework for testing HEVC under different packet loss rates, bandwidth restrictions, and network delay. As HEVC decoder is not robust against packet losses (it crashes if packets are missing) their framework decodes the complete bitstream and then overrides the areas corresponding to the missing packets. Several works propose mechanisms for protecting video streaming over vehicular networks although quality evaluation refers to percentage of lost packets [4]. In [5] the authors do a thorough research of protection of content delivery in vehicular environments searching for the best packet size in order to maximize throughput. They use different FEC techniques to protect data and offer results in terms of packet arrival ratio and file transfer time.

Our work differs in some aspects from the cited research works. The main difference is that our work combines several features of the cited works into one research. And also it includes features that are not included in previous works. In our work we have used real maps to design our vehicular scenarios. We have not used synthetic losses but the real losses obtained by simulations of vehicles moving through the scenario. We have used HEVC reference software with some

modifications in order to make it robust against packet losses (avoiding crashes when packets are lost). So we have decoded the bitstream with missing packets directly with the reference software. We have used RaptorQ codes in order to protect the video stream and we have tuned them to test which configurations get better results. And not only statistics about percentage of recovered packets are provided, but also the quality of the real reconstructed video sequence is presented (in terms of PSNR versus the original sequence).

The rest of the paper is structured as follows. In Section 2 the three main components of the scenarios are presented: vehicular networks, HEVC, and RaptorQ codes. In Section 3 the framework where the tests have been done is explained. Section 4 explains the results obtained for the different configurations that we have tested. At last, in Section 5, several conclusions are drawn.

2. Components

In this section we describe the three integral parts of the scenarios of our research: vehicular networks, HEVC, and RaptorQ codes.

2.1. Vehicular Networks. In future, vehicles will be equipped with lots of sensors (nowadays they already are) which will be able to collect internal and external measurements. They will also be provided with a certain computing capability which will allow them to process information, and they will also carry communications equipment. These three elements will make Intelligent Transport Systems (ITS) possible, where vehicles will have the ability to communicate with each other and with infrastructure in an intelligent way. This capacity of vehicles to communicate with each other and with a fixed network will bring both vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) networks. These networks will be used by applications regarding areas like people safety, fuel consumption savings, reduction of CO_2 emissions, infotainment, and so forth. Video streaming will be used by diverse types of applications like digital entertainment, Video On Demand (VOD), tourist information, contextual advertising, traffic flow density, and other regarding safety issues like emergency video call and so forth. Vehicular networks will have many challenges due to their nature. The combination of wireless communications with the relative high speed of nodes, variability of routes, and obstacles disturbing wireless signal (buildings, other vehicles, etc.) will make them packet loss prone networks. In packet loss prone networks, video streaming applications, which produce big deals of data and have high bandwidth requirements, may decrease the network performance and may have to deal with high rates of packet loss. These big deals of data need to be efficiently compressed to diminish the bandwidth needed for streaming. So we will need the assistance of proficient video codecs.

2.2. HEVC. On January 2013, High Efficiency Video Coding [6] was agreed upon as the new video coding standard. By 2010, ITU-T Video Coding Experts Group (VCEG) and ISO/IEC Moving Pictures Experts Group (MPEG) joined their research efforts and constituted the Joint Collaborative Team on Video Coding (JCT-VC) in order to develop a new video coding standard that would improve the previous one, H.264/AVC (Advanced Video Coding) [7], and could keep pace with the growing resolutions and frame rates of new video contents. The new standard follows the same hybrid compression scheme as its predecessor but includes a good number of refinements and new features that nearly doubles its coding efficiency. Here we will explain one of the features of HEVC that we will use in our tests: slices. For a deeper insight into the standard, some of the members of JCT-VC provide a complete overview in [8].

Slices are components of an encoded video stream that were introduced in the previous video coding standard, H.264/AVC. They are coded fragments of a frame that can be independently decoded. This makes them especially useful in packet loss prone scenarios. If we encode each frame of a video sequence using only one slice and that slice is bigger than the network MTU, then we will have to divide the slice in several fragments which will travel in several network packets. If one of these packets gets lost, then the rest of the fragments of the slice will become completely useless, because a slice cannot be decoded if it is not complete. In this way the loss of a single packet implies the effective loss of the whole frame. But if we divide each frame into several slices and each slice is smaller than the network MTU, then we will be able to send each slice in one packet. The loss of a single packet would not imply the loss of the whole frame but only the loss of a slice because the rest of the packets could be independently decoded.

But there is a drawback in dividing a frame into several slices. As slices are independently decodable, prediction is not allowed farther away of the slice limits. For instance, spatial prediction using areas of other slices is not allowed. The same happens to motion vectors prediction that must remain inside slice limits. This causes a decrease in coding efficiency. For every slice we will have a slice header which will also introduce some overhead. And if we divide a frame into too many slices, then the size of each slice may be much smaller in comparison with other headers used for streaming (e.g., RTP) and this would introduce a considerable overhead. In this work we will study how HEVC behaves in video streaming over vehicular networks for different number of slices per frame with the aid of RaptorQ codes to deal with packet loss.

2.3. RaptorQ Codes. Raptor codes, invented by Shokrollahi [9, 10], are a type of Fountain codes and are based in Luby Transform (LT) codes [11]. Raptor codes are a Forward Error Correction (FEC) technology which implements application layer protection against network packet losses. RaptorQ codes are a new family of codes that provide superior flexibility, support for larger source block sizes, and better coding efficiency than Raptor codes. They have several features that make them an interesting technology. One of them is that they can encode (protect) and decode (restore) data with linear time. They can also add variable levels of protection to better suit the protection to the network characteristics

(e.g., packet loss ratio, maximum bandwidth, etc.). They have very good recovery properties, because they can completely recover the original data if they receive approximately the same amount of data than the original one, regardless of whether the received packets are original packets or repair packets. RaptorQ codes are very efficient and have small memory and processing requirements, so they can be used in a wide variety of devices (from smartphones to big servers). Raptor and RaptorQ codes have been standardized by IETF [12, 13] and are used in 3GPP Multimedia Broadcast Multicast Services (MBMS) for file delivery and streaming.

This is how RaptorQ operates. The RaptorQ encoder receives a data stream (source packets) during a specified protection period. These packets are put together in memory to form a source block. A 4-byte FEC trailer is added to each received packet. This trailer identifies the packet and the protection period to which it belongs. These FEC-protected packets are sent through the network. When the protection period finishes, the source block in memory is FEC-encoded into repair symbols which are placed into repair packets and sent through the network. At the receiver side, the RaptorQ decoder receives protected source packets and repair packets. Some of these packets may get lost or corrupted and the RaptorQ decoder will try to recover lost packets out of the group of source and repair packets correctly received.

Latency (and, in some cases, memory consumption) is the drawback of RaptorQ protection scheme. As the protection window increases, the delay grows. Live events or real-time applications like video conference will have to use short periods for the protection window to keep latency into reasonable limits. Other types of streaming applications like IPTV may tolerate wider protection windows (while keeping latency inside a reasonably interaction response time). And some other applications like Video On Demand can be much more flexible in enlarging the protection period which will provide more bandwidth efficiency.

3. Framework

In our tests, several tools and simulators have been used. For the construction of the vehicular scenario we have used the OpenStreetMap project [14]. The OpenStreetMap project is a public domain geographic data base, built upon contributions of volunteers of all around the world. It is a project like Wikipedia but with geographic data, where you can find varied information like streets (including the number of lanes and their direction), parks, squares, schools, bus stops, singular buildings, drugstores, rivers, and so forth. From this page we have downloaded a real map of the city of Kiev. This map (XML data) has to be converted in order to be handled by SUMO (simulation of urban mobility) [15]. SUMO is a traffic simulator which is well known by the scientific community. It is able to run traffic simulations including vehicles, traffic lights, crossroads, priorities, and so forth and allows defining characteristics like the size, acceleration, deceleration and maximum speed of vehicles, and gathering different statistics like fuel consumption and CO2 emissions. One of the tools which is included in SUMO is TraCI (Traffic Control Interface). By using this interface, a bidirectional communication is possible between SUMO and other applications. In our tests we will connect SUMO (which will run the simulation of vehicles mobility) with OMNeT++ [16] where the vehicular network will be tested.

OMNeT++ is not a simulator itself but a framework for the development of simulators. It is a public domain software and a lot of projects (which implement the actual simulators) are available for it. Two of these projects have been put together to provide a vehicular network simulator, MiXiM [17] and Veins [18]. MiXiM is a project which implements wireless communications both fixed and mobile. Veins adds some protocols which have been standardized for vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) wireless communications, like IEEE 802.11p [19] and the IEEE 1609 family of standards [20].

Inside this vehicular network simulator we have developed an application for driving the experiments. It basically allows the injection of a video stream and the insertion of background traffic to produce congestion in the network to simulate adverse conditions. It also gathers some statistics for subsequent analysis.

For the encoding and decoding of HEVC video we have used the HEVC reference software [21]. That software has been modified by us in order to generate RTP packets inside the bitstream. In addition we have modified the HEVC decoder to make it resilient to packet losses. We have used two different encoding modes, specified by the JCT-VC. On the one hand we have used All Intra (AI) mode, in which every frame of the video sequence is encoded as an I frame. An I frame is encoded wihtout using other frames as a reference, and therefore, it is independent of the rest of the frames. This is called intramode. An I frame exploits the spatial redundancy that exists inside the frame. On the other hand, we have used Low-Delay P (LP) mode. In this mode the first frame is encoded as an I frame and the rest of the frames are P (predictive) frames. P frames use other frames previously encoded (and decoded) as a reference to exploit temporal redundancy. They use motion estimation/compensation. This is called inter mode. Inter mode is more efficient than intramode; thus, better compression rates are obtained by LP encoding mode than by AI encoding mode. In LP mode, P frames depend on other frames used as a reference. This makes this mode more vulnerable to packet losses. In AI mode the loss of a slice will only affect one frame, but in LP mode, the loss of a slice will affect the frame which it belongs to and also the frames that use this frame as a reference. The frames that use these affected frames will also be damaged and so on. To stop this drift and try to alleviate packet losses (those that could not be recovered with RaptorQ codes) we have used the intra-refresh mechanism in LP encoded streams. This mechanism consists basically in inserting an I frame every 32 frames. If all the slices that belong to this frame arrive to their destination (or equivalently, RaptorQ codes are able to recover them) the sequence is refreshed and the propagation of errors ends.

For the protection of the bitstream with RaptorQ codes we have used the Qualcomm (R) RaptorQ (TM) Evaluation Kit [22]. RaptorQ software has different options to better tune

it and suit it to fit your needs or preferences. For instance, you can specify the amount of protection to add to the bitstream (i.e., 10%, 20%, etc.). The bigger the amount of protection you add, the higher the probability of successful recovery of lost packets will be, but also the higher the bandwidth required for the transmission will become. You can also adjust the length of the temporal window which will divide the bitstream in fractions in order to generate FEC packets. You can also set the symbol size and the repair packet size. An adequate symbol size can make the encoding and decoding of FEC data more computationally efficient and also reduce the amount of memory required for these computations. The final aim of this work is to evaluate how HEVC and RaptorQ codes behave and how they can collaborate to protect video transmissions in vehicular networks and determine which are the most suitable configurations of both in each situation. This is a necessary step to propose adaptive mechanisms that can optimize resources and provide error resilience techniques that assure a good quality of experience in video streaming via vehicular networks.

For the performance of the tests we have followed these steps. Once the scenario has been implemented, we have encoded a raw video sequence at different number of slices per frame producing several HEVC bitstreams (in RTP format). Each one of these bitstreams has been FEC encoded with several configurations using RaptorQ. Protected sequences have been used to run simulations. Each simulation produces a file with several statistics (including the packet loss ratio) and a file with the received packets. This file is FEC decoded by means of RaptorQ in order to try to recover lost packets and to produce an HEVC file (in RTP format). This RTP file is decoded by HEVC decoder and finally the reconstructed raw video sequence is compared to the original one in order to calculate PSNR and evaluate final video quality.

In next section we will give some extra details about tests and we will analyze the most relevant results obtained.

4. Experiments and Results

The study case is based on the scenario shown in Figure 1. It is an area of 2000 m × 2000 m of the city of Kiev. In it we can find a long avenue that crosses that area from north to south. Along the avenue, 3 road side units (RSUs) have been positioned, named A, B, and C in the figure. These RSUs will transmit the video sequence simultaneously (in a synchronized way). The coverage radius of the wireless devices is 500 m. RSUs A and B have a small area where their signals overlap. And RSUs B and C have a small shaded area where neither of the two can reach. Therefore we have three different types of areas regarding transmission: areas where a vehicle can receive the data from only one RSU, one area where the vehicle receives the signal from two RSUs (A and B), and one area where signal is momentarily lost (between B and C coverage areas). A total of 50 vehicles have been inserted into the scenario, driving in different routes. Those vehicles send a beacon every second through the control channel (following IEEE 1609.4 multichannel operations).

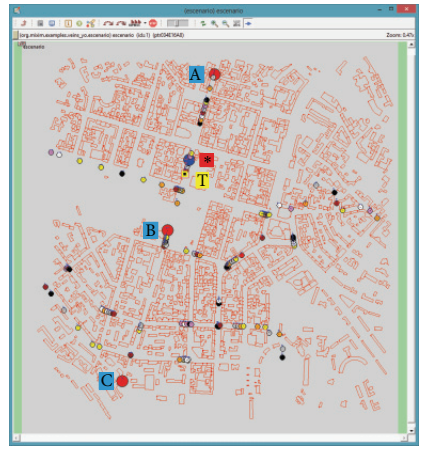


Figure 1: Vehicular network scenario in OMNeT++ (red circles A, B, C = RSUs//blue circle ∗ = video client//yellow square T = background traffic source//small circles = other vehicles//red rectangles = buildings).

We also have a vehicle driving near the video client that can act as a background traffic source (labeled as T in Figure 1), sending packets through the wireless network at different packets per second (pps) rates. The video client (marked in the figure as ∗) will experience isolated packet losses (mainly due to background traffic) and bursty packet losses (around the limits of RSUs coverage). RSUs send periodically through the control channel advertisements of the video service that they offer, indicating the service channel used for the video stream. The video client receives that invitation and commutes to the specified service channel in order to receive the video stream.

4.1. HEVC Evaluation. Now we present the evaluation of the video sequence behavior when it is encoded at a different number of slices per frame in both encoding modes used (AI and LP).

The sequence chosen for the tests is RaceHorses, one of the test sequences used by JCT-VC for HEVC evaluation in common test conditions [23]. It has a resolution of 832x480 pixels and a frame rate of 30 frames per second. We have encoded it at 1, 2, 4, 8, 13, and 26 slices per frame (slc/frm). For the encoding process we have used a value of 37 for the quantization parameter (QP). For that value we obtain a mean PSNR value of 32.12 dB for AI mode (1 slc/frm) and a value of 30.19 dB for LP mode (1 slc/frm). Video quality is higher in AI mode but bitrate in LP mode is much lower. As predictions cannot cross slice boundaries, when we split each frame into several slices we are reducing coding efficiency because we cannot use information of nearby areas if they do not belong to the slice. An example of this penalization is that slices cannot use intraprediction between slices (in AI mode) and slices cannot use predictions for motion vectors (in LP mode). Figure 2 shows the percentage of increment

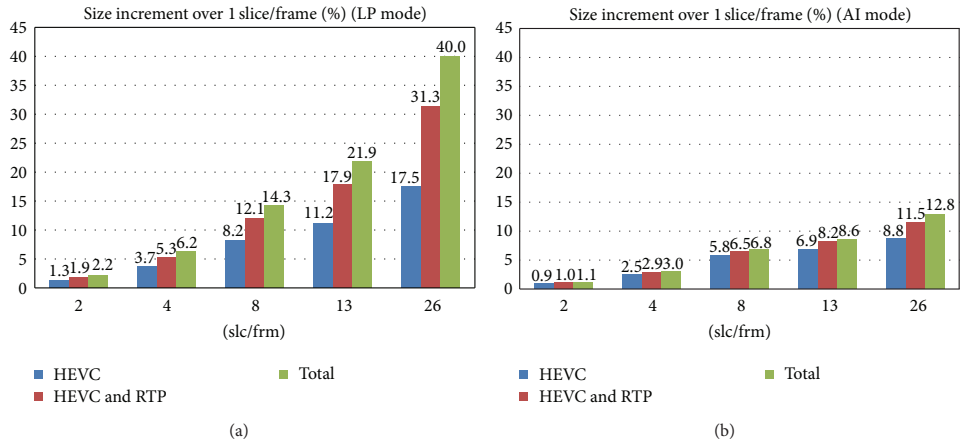
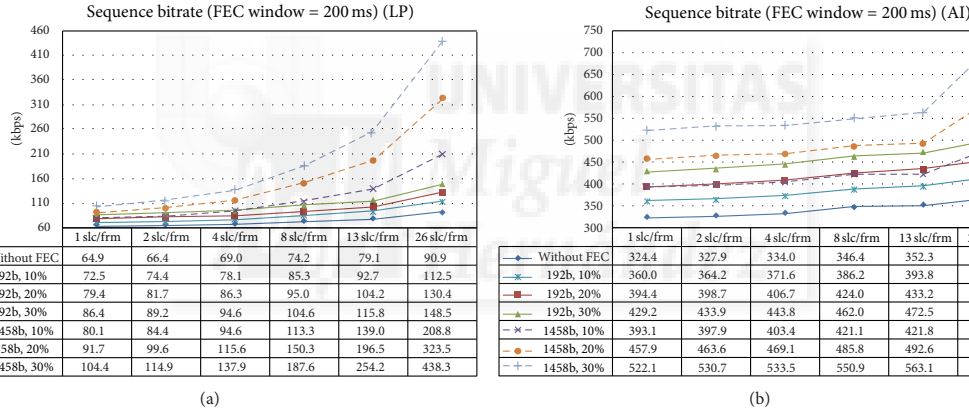


Figure 2: Percentage of bitrate increase (without FEC protection) for different number of slices per frame. (a) LP mode. (b) AI mode. (HEVC raw bitstream//HEVC + RTP header//HEVC + RTP header + fragmentation header.)



	1 slc/frm	2 slc/frm	4 slc/frm	8 slc/frm	13 slc/frm	26 slc/frm
Without FEC	64.9	66.4	69.0	74.2	79.1	90.9
192b, 10%	72.5	74.4	78.1	85.3	92.7	112.5
192b, 20%	79.4	81.7	86.3	95.0	104.2	130.4
192b, 30%	86.4	89.2	94.6	104.6	115.8	148.5
1458b, 10%	80.1	84.4	94.6	113.3	139.0	208.8
1458b, 20%	91.7	99.6	115.6	150.3	196.5	323.5
1458b, 30%	104.4	114.9	137.9	187.6	254.2	438.3

(a)

	1 slc/frm	2 slc/frm	4 slc/frm	8 slc/frm	13 slc/frm	26 slc/frm
Without FEC	324.4	327.9	334.0	346.4	352.3	365.9
192b, 10%	360.0	364.2	371.6	386.2	393.8	413.6
192b, 20%	394.4	398.7	406.7	424.0	433.2	457.7
192b, 30%	429.2	433.9	443.8	462.0	472.5	502.0
1458b, 10%	393.1	397.9	403.4	421.1	421.8	483.8
1458b, 20%	457.9	463.6	469.1	485.8	492.6	598.5
1458b, 30%	522.1	530.7	533.5	550.9	563.1	713.3

(b)

Figure 3: Bitrate (kbps) without FEC protection and with FEC protection for different symbol sizes, different protection windows, and different number of slices per frame (including RTP and fragmentation headers). (a) LP mode. (b) AI mode.

of the encoded sequence size at different number of slices per frame compared to 1 slc/frm. It shows the percentage of increment of HEVC raw bitstream and also the percentage of increment after adding the RTP headers to each slice. If the size of one slice (including its RTP header) is greater than the network MTU, then it will be divided into some fragments. If one of the fragments of a slice gets lost, then the whole slice will be discarded because it will be undecodable. So the rest of the fragments of the slice are automatically discarded. We identify every fragment of a slice with a header in order to know if a slice has received all its fragments. In Figure 2, data labeled as "TOTAL" shows the bitrate increment with respect to 1 slc/frm when both RTP and fragmentation headers are included. As slices generated by LP mode are much smaller than slices generated by AI mode (because of LP mode coding efficiency), the overhead introduced by RTP and fragmentation headers is much greater for LP mode, and consequently the same happens to the total percentage of increment in the bitstream. For example, encoding at 13 slc/frm increases the bitstream around a 20%, and encoding at 26 slc/frm increases it around a 40% for LP mode. But for the same number of slices per frame, in AI mode the increments are around 8.6% and 12.8%, respectively.

In Figure 3, if you look at the curve labeled "without FEC," you can see the bitrate value (not the percentage of increment) of the sequence for LP and AI modes. In LP mode the bitrates range from 64.9 kbps (1 slc/frm) to 90.9 kbps (26 slc/frm). In AI mode the bitrates range from 324.4 kbps (1 slc/frm) to 365.9 kbps (26 slc/frm). As we stated before in this section, the bitrate for mode LP is much lower than for AI mode; specifically, at 1 slice per frame the bitrate of LP mode is 5 times lower than AI's one.

A factor that can be of great relevance when protecting data is the proportion of network packets (fragments) with

Table 1: Mean proportion of fragments (network packets) for every RTP packet (slice).

Fragments/RTP	1 sl	2 sl	4 sl	8 sl	13 sl	26 sl
LP mode	1.87	1.37	1.21	1.02	1.00	1.00
AI mode	7.90	4.29	2.38	1.51	1.07	1.00

Table 2: Packet rate (packets per second) for LP mode without FEC protection and with FEC protection at 30% of redundancy for different symbol sizes, different protection windows, and different number of slices per frame.

Packets/sec.	1 sl	2 sl	4 sl	8 sl	13 sl	26 sl
Without FEC	56.1	82.0	134.6	244.9	391.3	780.0
192 b, 133 ms	75.4	102.7	156.2	270.0	420.7	824.6
192 b, 200 ms	73.8	100.5	155.0	268.7	419.6	822.7
192 b, 250 ms	73.1	100.1	154.6	268.4	419.0	821.5
192 b, 333 ms	72.9	99.8	154.2	267.7	418.4	820.7
192 b, 500 ms	72.1	99.2	153.3	266.5	417.0	821.5
192 b, 1000 ms	71.2	97.9	152.0	265.2	415.5	820.9
1458 b, 133 ms	83.2	116.3	183.0	320.4	509.1	1016.7
1458 b, 200 ms	82.9	114.9	181.3	321.7	509.8	1015.1
1458 b, 250 ms	81.8	114.5	180.8	321.6	509.1	1010.0
1458 b, 333 ms	81.8	114.0	180.7	320.0	507.9	1008.9
1458 b, 500 ms	80.5	113.7	179.1	317.8	504.3	1003.9
1458 b, 1000 ms	79.5	111.6	176.2	313.2	498.1	1013.9

respect to RTP packets (each RTP packet includes one slice). As stated before, if we have many fragments for one RTP packet, then the probability of losing this RTP packet increases, because the real loss of only one of the fragments will render the RTP packet useless. And this will result in the effective loss of the rest of the fragments of that RTP packet. But if we have one fragment per RTP packet, then the loss of one fragment will not affect the rest of packets. When the proportion tends to 1 every lost packet which can be recovered by RaptorQ is contributing to improve the final quality of the reconstructed video. When the proportion moves far away from 1 the recovery of packets by RaptorQ does not always directly turn into an improvement on video quality.

The mean proportion of fragments per RTP packet is shown in Table 1. Data show that when using AI mode and few slices per frame the proportion is far from 1, and this indicates that packet recovery will not be as productive as for LP mode. Data of Table 1 will be different if we encode other video sequences with larger (or smaller) resolutions or with different values for QP parameter or with different values for intra-refresh period. The conclusion is that before protecting video streams with RaptorQ codes, we should take into consideration this proportion in order to better decide the suitable number of slices per frame that will take a greater advantage of FEC protection.

4.2. RaptorQ Codes Setup. For the evaluation of RaptorQ and determining which configurations are most suitable, we have protected each of the generated bitstreams with different setups for RaptorQ. 6 different sizes for the temporal window have been used (133, 200, 250, 333, 500, and 1000 milliseconds), 3 different levels of protection have been assigned (10%, 20% and 30%), and 2 different combinations of symbol size and repair packet size have been used (a small symbol size of 192 bytes together with a repair packet size of 7 and a big symbol size of 1458 bytes together with a repair packet size of 1).

First of all we have analyzed the overhead generated by the protection added with RaptorQ codes. In addition to the global bitrate increment we have measured the increment in the packet rate (which is caused by the addition of repair packets). In previous works we found out that in vehicular networks packet losses are more influenced by packet rate than by packet size, so if we do not keep the extra packets per second low, then the solution to our problem (adding extra repair packets) can become the problem of the network.

Table 2 shows the number of packets per second in the transmission of the LP encoded sequence without any FEC protection and also with FEC protection with a redundancy of 30% for different symbol sizes and different temporal windows. When we use a big symbol size, more packets per second are generated than when we use a small symbol size. This gets worse as the number of slices increases. It can also be seen that varying the temporal window does not change significantly the packet rate.

Table 3 shows the packet rate without FEC protection and by using FEC protection with a symbol size of 192 bytes and a temporal window of 200 ms. If we compare the number of packets per second for LP and AI modes without using FEC we can see the correlation with Table 1. When we encode the video sequence at 26 slc/frm, we have a proportion of 1.00 fragment per RTP packet for both LP and AI modes. This means that every single RTP packet is smaller than the MTU

Table 3: Packet rate (packets per second) without FEC protection and with FEC protection for a symbol size of 192 bytes and a protection window of 200 ms for different coding modes, different percentages of redundancy, and different number of slices per frame.

Packets/sec.	1 sl	2 sl	4 sl	8 sl	13 sl	26 sl
(LP) w/o FEC	56.1	82.0	134.6	244.9	391.3	780.0
(LP) 10%	63.6	89.9	143.3	254.9	402.0	795.4
(LP) 20%	68.5	95.0	148.9	261.9	410.8	808.9
(LP) 30%	73.8	100.5	155.0	268.7	419.6	822.7
(AI) w/o FEC	237.0	257.3	285.3	362.8	417.6	780.0
(AI) 10%	265.0	285.6	314.8	392.9	449.3	815.0
(AI) 20%	290.2	311.0	340.6	421.1	478.2	847.6
(AI) 30%	316.0	336.8	367.7	449.4	507.4	880.7

and the packet rate can be directly calculated by multiplying the frame rate (30 fps) by the number of slices per frame (26 slc/frm); this is 780.0 pps. At the opposite side of Table 3, if we look at AI mode without using FEC encoded at 1 slc/frm, we can see that the packet rate (237.0 pps) is very far from the multiplication of the frame rate (30 fps) by the number of slices per frame (1 slc/frm). At 1 slc/frm AI mode and LP mode produce very different packet rates. Observing FEC-protected data it can be seen that when fewer slices per frame are used the proportion of extra packets per second is greater.

Figure 3 shows the bitrate (kbps) of the encoded sequences in modes LP and AI without FEC protection and with FEC protection for three different levels of redundancy and two symbol sizes. Selecting 192 bytes as the symbol size leads to much lower bitrates. This is more emphasized for LP coding mode.

4.3. OMNeT++ Tests. In this section we will present the tests performed in the simulations. For the experiments we have used the framework previously depicted, using SUMO, OMNeT++, MiXiM, and Veins.

In simulations we connect SUMO and OMNeT++ via TraCI. SUMO tells OMNeT++ the position of every vehicle at every instant and OMNeT++ (using MiXiM and Veins) runs the simulation of the vehicular network. Video sequences which have been previously protected with RaptorQ codes are transmitted from the 3 RSUs to the video receiver. At the end of the simulation a file is generated with some statistics, including the percentage of packets lost. Another file is also generated including the packets received by the vehicle. This file is processed using RaptorQ decoder in order to generate a file with RTP packets, trying to recover the lost packets. The output of this process is passed to the HEVC decoder which will try to restore the video sequence. After this process we compute the PSNR value for the reconstructed sequence.

We have run simulations for the following combinations: both modes of encoding (AI and LP), different number of slices per frame (1, 4, 8, and 13 slc/frm), and three protection levels (10%, 20%, 30%). For all these combinations we have run tests injecting background traffic at four different rates (0, 30, 240, and 390 pps).

In areas of full coverage, packet losses are due to background traffic. Here we encounter isolated losses. On the contrary, in areas near the limits of the RSUs coverage, packet losses are bursty because the signal is completely lost for some period. For isolated losses RaptorQ codes do a good job in recovering lost packets. Tables 4 and 5 show the total percentage of network packet loss, the percentage of RTP packet loss after recovery, and the difference in PSNR of the reconstructed video for LP and AI modes, respectively (for areas of good coverage). As it can be seen, RaptorQ can recover a high percentage of network packets and the result is that only a small percentage of RTP packets are lost. This is not true for AI mode and few slices per frame (1 slc/frm and 2 slc/frm), but this is the expected behavior taking into account that, as we stated before, in these configurations the proportion of fragments per RTP packet is high. We can observe from those tables that AI mode is inherently more error resistant than LP mode. This is also the expected behavior because in LP mode reference pictures with incorrect data propagate errors and in AI mode errors do not propagate. Some techniques like unequal error protection methods could be useful to introduce different levels of protection regarding the importance of the video packets (I or P frames) in the final quality of the reconstructed sequence. Regarding areas near the limits of RSUs coverage, we can state that RaptorQ is not able to deal with bursty losses

Table 4: Total percentage of network packet loss, percentage of RTP packet loss after recovery, and difference in PSNR of the reconstructed video, for a background traffic of 390 pps and a level of protection of 30%, for areas with good signal coverage. LP encoding mode.

slc/frm	Measurement	192 b	1458 b
1 sl	TOTAL loss (%)	11.28	10.89
1 sl	RTP loss (%)	1.28	0.18
1 sl	PSNR diff (dB)	0.99	0.18
4 sl	TOTAL loss (%)	13.55	14.36
4 sl	RTP loss (%)	1.84	0.62
4 sl	PSNR diff (dB)	2.85	0.75
8 sl	TOTAL loss (%)	14.59	15.05
8 sl	RTP loss (%)	1.47	0.09
8 sl	PSNR diff (dB)	1.90	0.11
13 sl	TOTAL loss (%)	13.97	13.25
13 sl	RTP loss (%)	0.29	0.00
13 sl	PSNR diff (dB)	0.52	0.00

Table 5: Total percentage of network packet loss, percentage of RTP packet loss after recovery and difference in PSNR of the reconstructed video, for a background traffic of 390 pps and a level of protection of 30%, for areas with good signal coverage. AI encoding mode.

slc/frm	Measurement	192 b	1458 b
1 sl	TOTAL loss (%)	19.07	19.10
1 sl	RTP loss (%)	14.81	16.94
1 sl	PSNR diff (dB)	2.20	2.52
4 sl	TOTAL loss (%)	18.34	18.25
4 sl	RTP loss (%)	6.76	4.17
4 sl	PSNR diff (dB)	1.32	0.81
8 sl	TOTAL loss (%)	16.65	16.13
8 sl	RTP loss (%)	2.56	0.93
8 sl	PSNR diff (dB)	0.71	0.23
13 sl	TOTAL loss (%)	15.06	14.72
13 sl	RTP loss (%)	0.46	0.21
13 sl	PSNR diff (dB)	0.15	0.06

because within the protection period very few packets are received and there is not enough data to carry out a recovery. This problem could be addressed with the introduction of techniques like interleaving, where, with the trade-off of the introduction of some delay, bursty losses can be easily converted to isolated losses (where RaptorQ can get good percentages of recovery). We can see that a symbol size of 1458 bytes provides better recovery results but if we take into consideration data from Figure 3, then the overhead introduced is not bearable. At last, from the very specific conditions of our tests we can state that the optimum number of slices per frame is 8, which produces the best trade-off between the introduced overhead (both packets per second and total bitrate) and the percentage of recovery and the final video quality.

5. Conclusions

In this work we have analyzed the protection of video delivery in vehicular networks. The video encoder selected for the tests is the new emerging standard HEVC. To protect the video stream we have used RaptorQ codes. We have first analyzed the behavior and performance of HEVC for two coding modes (AI and LP) and for different number of slices per frame. Then we have protected the encoded sequences by means of RaptorQ with several configurations and observed the effects of this selection. We have seen that varying HEVC and RaptorQ parameters leads to very different situations, regarding total bitrate and packet rate. At last we have run simulations in a vehicular environment to measure the ability of RaptorQ in protecting video packets in this type of scenarios. The reduction in the number of lost packets because of RaptorQ codes recovery properties has been presented as well as the quality of the decoded video reconstructions. As a general conclusion we can state that there are a lot of parameters that can be fine-tuned to adapt the video protection to the requirements of the specific network conditions (bandwidth, packet loss ratio, isolated/bursty losses, etc.) and to the requirements of the specific application (encoding mode, level of quality, resolution of video sequence, etc.). So there is not a general formula which will fit into all situations to provide the best level of protection. On the contrary, a previous evaluation of the real conditions and user preferences is mandatory.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This research was supported by the Spanish Ministry of Education and Science under Grant TIN2011-27543-C03-03, the Spanish Ministry of Science and Innovation under Grant TIN2011-26254, and Generalitat Valenciana under Grant ACOMP/2013/003.

References

[1] J.-R. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan, and T. Wiegand, "Comparison of the coding efficiency of video coding standards-including high efficiency video coding (HEVC)," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1669–1684, 2012.

[2] R. Garcia and H. Kalva, "Subjective evaluation of HEVC in mobile devices," in *Multimedia Content and Mobile Devices*, vol. 8667 of *Proceedings of SPIE*, February 2013.

[3] J. Nightingale, Q. Wang, and C. Grecos, "HEVStream: a framework for streaming and evaluation of high efficiency video coding (HEVC) content in loss-prone networks," *IEEE Transactions on Consumer Electronics*, vol. 58, no. 2, pp. 404–412, 2012.

[4] J. Park, U. Lee, and M. Gerla, "Vehicular communications: Emergency video streams and network coding," *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 57–68, 2010.

[5] C. T. Calafate, G. Fortino, S. Fritsch, J. Monteiro, J.-C. Cano, and P. Manzoni, "An efficient and robust content delivery solution for IEEE 802.11p vehicular environments," *Journal of Network and Computer Applications*, vol. 35, no. 2, pp. 753–762, 2012.

[6] B. Bross, W.-J. Han, J.-R. Ohm et al., "High Efficiency Video Coding (HEVC) text specification draft 10," Tech. Rep. JCTVC-L1003, Joint Collaborative Team on Video Coding (JCT-VC), Geneva, Switzerland, 2013.

[7] ITU-T and ISO/IEC JTC 1, "Advanced Video Coding for Generic Audiovisual Services," ITU-T Rec. H.264 and ISO/IEC 14496-10 (AVC) version 16, 2012.

[8] G. J. Sullivan, J. Ohm, W. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[9] A. Shokrollahi, "Raptor codes," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2551–2567, 2006.

[10] M. A. Shokrollahi and M. Luby, "Raptor codes," *Foundations and Trends in Communications and Information Theory*, vol. 6, no. 3-4, pp. 213–322, 2009.

[11] M. Luby, "LT codes," in *Proceedings of the 34rd Annual IEEE Symposium on Foundations of Computer Science*, pp. 271–280, November 2002.

[12] M. Luby, A. Shokrollahi, M. Watson, and T. Stockhammer, "Raptor forward error correction scheme for object delivery," in *IETF RMT Working Group, Work in Progress*, RFC 5053, 2007.

[13] M. Luby, A. Shokrollahi, M. Watson, T. Stockhammer, and L. Minder, "RaptorQ Forward Error Correction Scheme for Object Delivery," IETF RMT Working Group, Work in Progress, RFC 6330, 2011.

[14] OpenStreetMap, http://www.openstreetmap.org/.

[15] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO—simulation of urban mobility," *International Journal on Advances in Systems and Measurements*, vol. 5, no. 3-4, pp. 128–138, 2012.

[16] OMNeT++, http://www.omnetpp.org/.

[17] MiXiM, http://mixim.sourceforge.net/.

[18] C. Sommer, R. German, and F. Dressler, "Bidirectionally coupled network and road traffic simulation for improved IVC analysis," *IEEE Transactions on Mobile Computing*, vol. 10, no. 1, pp. 3–15, 2011.

[19] "IEEE Standard for Information technology—Local and metropolitan area networks—Specific requirements—Part: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Wireless Access in Vehicular Environments," IEEE Std 802.11p-2010, pp. 1–51, 2010.

[20] "IEEE standard for Wireless Access in Vehicular Environments (WAVE)–multi-channel operation," Tech. Rep. IEEE Std 1609.4-2010, 2011.

[21] H M Reference Software vers. 9.0, https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-9.0/.

[22] Qualcomm (R) RaptorQ (TM) Evaluation Kit, http://www.qualcomm.com/raptor-evaluation-kit.

[23] F. Bossen, "Common test conditions and software reference configurations," Tech. Rep. JCT VC-L1100, Joint Collaborative Team on Video Coding (JCT-VC), Geneva, Switzerland, 2013.

Bibliografía

[1] G. Karagiannis, O. Altintas, E. Ekici, G. Heijenk, B. Jarupan, K. Lin, and T. Weil. Vehicular Networking: A Survey and Tutorial on Requirements, Architectures, Challenges, Standards and Solutions. *Communications Surveys Tutorials, IEEE*, 13(4):584–616, November 2011.

[2] Sherali Zeadally, Ray Hunt, Yuh-Shyan Chen, Angela Irwin, and Aamir Hassan. Vehicular Ad Hoc Networks (VANETS): Status, Results, and Challenges. *Telecommunication Systems*, 50(4):217–241, August 2012.

[3] J.R. Wootton, A. García-Ortiz, and S.M. Amin. Intelligent Transportation Systems: A Global Perspective. *Mathematical and Computer Modelling*, 22(4):259–268, August 1995.

[4] Lino Figueiredo, Isabel Jesus, J.A.Tenreiro Machado, Jose Rui Ferreira, and J.L.Martins de Carvalho. Research Issues in Intelligent Transportation Systems. In *Control Conference (ECC), 2001 European*, pages 301–306, September 2001.

[5] A.B. Nkoro and Y.A. Vershinin. Current and Future Trends in Applications of Intelligent Transport Systems on Cars and Infrastructure. In *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*, pages 514–519, October 2014.

[6] Mohammed Ghanbari. *Standard Codecs: Image Compression to Advanced Video Coding*. Institution Electrical Engineers, ISBN: 978-0-86341-964-5, 481 pp., 2011.

[7] Wen Gao and Siwei Ma. *Advanced Video Coding Systems*. Springer Publishing Company, ISBN: 978-3-319-14243-2, 239 pp., 2014.

[8] G.J. Sullivan, J. Ohm, Woo-Jin Han, and T. Wiegand. Overview of the High Efficiency Video Coding (HEVC) Standard. *Circuits and Systems for Video Technology, IEEE Transactions on*, 22(12):1649–1668, 2012.

[9] ITU-T and ISO/IEC JTC 1. Advanced Video Coding for Generic Audiovisual Services. *ITU-T Rec. H.264 and ISO/IEC 14496-10 (AVC) version 16, 2012*.

[10] HM Reference Software vers. 9.0, https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-9.0.

[11] Frank Bossen. Common test conditions and software reference configurations. Technical Report JCTVC-L1100, Joint Collaborative Team on Video Coding (JCT-VC), Geneva (Switzerland), January 2013.

[12] OpenMP Application Program Interface, version 3.1, http://www.openmp.org. *OpenMP Architecture Review Board.*, 2011.

[13] OpenStreetMap, http://wwww.openstreetmap.org.

[14] Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker. Recent development and applications of SUMO - Simulation of Urban MObility. *International Journal On Advances in Systems and Measurements*, 5(3&4):128–138, December 2012.

[15] TraCI - Traffic Control Interface, http://sourceforge.net/apps/mediawiki/sumo/index.php?title=TraCI.

[16] OMNeT++ Discrete Event Simulator, http://www.omnetpp.org.

[17] MiXiM - Mixed Simulator, http://mixim.sourceforge.net.

[18] Christoph Sommer, Reinhard German, and Falko Dressler. Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis. *IEEE Transactions on Mobile Computing*, 10(1):3–15, January 2011.

[19] IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Wireless Access in Vehicular Environments. *IEEE Std 802.11p-2010*, pages 1–51, July 2010.

[20] IEEE Standard for Wireless Access in Vehicular Environments (WAVE)– Multi-channel Operation. *IEEE Std 1609.4-2010 (Revision of IEEE Std 1609.4-2006)*, pages 1–89, 2011.

[21] Qualcomm (R) RaptorQ (TM) Evaluation Kit, http://www.qualcomm.com/solutions/multimedia/media-delivery/raptor-evaluation-kit.