
This is the **published version** of the bachelor thesis:

Andreu Vidal, Manel. *Evaluación e Integración de Algoritmos de Compresión Entrópica en estructura k^2 -raster para Imágenes Hiperespectrales*. Treball de Final de Grau (Universitat Autònoma de Barcelona), 2026 (Enginyeria Informàtica)

This version is available at <https://ddd.uab.cat/record/334421>

under the terms of the  license.

Evaluación e Integración de Algoritmos de Compresión Entrópica en estructura k^2 -raster para Imágenes Hiperespectrales

Manel Andreu Vidal

Resum— La transmissió i emmagatzematge d'imatges hiperespectrals exigeixen algorismes de compressió sense pèrdues que permetin un accés aleatori àgil. Aquest treball integra i avalua codificadors entròpics (Golomb, Rice, Elias i Huffman) sobre l'estructura k^2 -raster. Mitjançant un mecanisme de fites a baix nivell, es resol la limitació seqüencial d'aquests codis. Els resultats revelen el col·lapse de Huffman davant l'alta variància, la latència gairebé nul·la de PForDelta, i la màxima eficiència en memòria de Simple16.

Paraules clau— Compresió sense pèrdues, k^2 -raster, codificació entròpica, imatges hiperespectrals, accés aleatori.

Resumen— La transmisión y almacenamiento de imágenes hiperespectrales exigen algoritmos de compresión sin pérdidas que permitan un acceso aleatorio ágil. Este trabajo integra y evalúa codificadores entrópicos (Golomb, Rice, Elias y Huffman) sobre la estructura k^2 -raster. Mediante un mecanismo de hitos a bajo nivel, se resuelve la limitación secuencial de estos códigos. Los resultados revelan el colapso de Huffman ante la alta varianza, la latencia casi nula de PForDelta, y la máxima eficiencia en memoria de Simple16.

Palabras clave— Compresión sin pérdidas, k^2 -raster, codificación entrópica, imágenes hiperespectrales, acceso aleatorio.

Abstract— The transmission and storage of hyperspectral images demand lossless compression algorithms that enable fast random access. This project integrates and evaluates entropy encoders (Golomb, Rice, Elias, and Huffman) on the k^2 -raster structure. Using a low-level milestone mechanism, the sequential limitation of these codes is resolved. Results reveal Huffman's collapse due to high variance, PForDelta's near-zero latency, and the maximum memory efficiency of Simple16.

Keywords— Lossless compression, k^2 -raster, entropy encoding, hyperspectral images, random access.



estándares internacionales exigen herramientas de compresión sin pérdidas (*lossless*) que no corrompan las firmas espectrales originales.

1 INTRODUCCIÓN

LA transmisión y el almacenamiento de datos procedentes de la observación de la Tierra constituyen uno de los mayores desafíos actuales en la ingeniería aeroespacial y de telecomunicaciones.

Los sensores hiperespectrales capturan información a través de cientos de bandas continuas, generando “cubos de datos” de enorme riqueza analítica pero de volúmenes masivos. Dado el alto valor científico de estas capturas, los

Históricamente, los enfoques tradicionales de compresión en bloque han presentado una limitación severa: la falta de acceso aleatorio (*random access*). Para consultar un único píxel, el sistema obliga a descomprimir secuencialmente grandes volúmenes de datos en la memoria RAM. Esta ineficiencia ha motivado la adopción de las Estructuras de Datos Compactas (CDS), destacando el k^2 -raster como solución optimizada para matrices espaciales. No obstante, el rendimiento de esta estructura depende de cómo se codifiquen internamente sus matrices diferenciales de máximos y mínimos.

En este contexto, el propósito de este Trabajo de Fin de

- E-mail de contacto: Manel.AndreuV@autonoma.cat
- Menció realizada: Tecnologies de la Informació
- Trabajo tutorizado por: Hing Fai Kevin Chow (DEIC)
- Curso 2025/26

Grado consiste en integrar y evaluar experimentalmente diversas familias de codificación entrópica sobre la estructura de datos compacta k^2 -raster. Se busca optimizar la compresión sin pérdidas de imágenes hiperespectrales garantizando la capacidad de acceso aleatorio, de forma que sea posible consultar valores puntuales sin necesidad de descomprimir la totalidad de la información.

Para lograr llevar a cabo este propósito, el desarrollo del proyecto contempla varias metas técnicas específicas encaenadas. En primer lugar, se estudia la partición espacial de la estructura y el comportamiento de los códigos de longitud variable frente a enfoques alineados a palabra (*word-aligned*). A partir de este análisis, se programan en lenguaje C y se integran nativamente en el motor base los codificadores estadísticos de Golomb, Rice, la familia Elias (Gamma, Delta y Fano) y Huffman. Para resolver el acceso secuencial de estos algoritmos, se diseña e incorpora un sistema de punteros de sincronización (*milestones*) que posibilita la extracción directa de píxeles individuales con una latencia controlada. Finalmente, se desarrolla un marco de validación en Python para orquestar la extracción de métricas de rendimiento, lo que permite contrastar la eficiencia espacio-temporal de las nuevas implementaciones frente a los estándares clásicos e identificar sus limitaciones ante datos de alta varianza.

2 ESTRUCTURA DEL INFORME

Para abordar los objetivos planteados, esta memoria se ha organizado de la siguiente forma. Tras esta primera contextualización, la Sección 3 expone el estado del arte y el marco teórico, detallando los fundamentos de la compresión hiperespectral, la topología del k^2 -raster y el funcionamiento analítico de las familias de codificación evaluadas. A continuación, la Sección 4 documenta la metodología y el proceso de ingeniería, justificando la integración en lenguaje C, la adaptación del sistema de acceso aleatorio y la construcción del entorno automatizado de pruebas. Posteriormente, la Sección 5 presenta la exposición de resultados mediante el análisis de las métricas extraídas, donde se discute la eficiencia espacial, se identifican las barreras arquitectónicas de ciertos algoritmos y se analiza el rendimiento de latencia. Por último, la Sección 6 sintetiza las conclusiones principales del estudio y valora el grado de cumplimiento de los objetivos.

3 ESTADO DEL ARTE

La transmisión y el almacenamiento de datos procedentes de la observación de la Tierra constituyen uno de los mayores desafíos actuales en la ingeniería aeroespacial y de telecomunicaciones. Para abordar este problema, la comunidad científica ha desarrollado diversas aproximaciones algorítmicas que buscan reducir la huella espacial de la información sin penalizar los tiempos de acceso computacional. En esta sección se exponen los fundamentos teóricos de las imágenes hiperespectrales, las Estructuras de Datos Compactas (CDS) y las principales familias de compresión empleadas en la literatura para la codificación de enteros.

3.1. Compresión de imágenes hiperespectrales

Los sensores hiperespectrales montados en satélites y plataformas aerotransportadas capturan la radiación electromagnética reflejada por la superficie terrestre a través de cientos de bandas espectrales contiguas. A diferencia de las imágenes RGB convencionales, que constan de tres canales discretos, una imagen hiperespectral forma un “cubo de datos” tridimensional, donde dos dimensiones representan las coordenadas espaciales (X, Y) y la tercera dimensión representa la profundidad espectral (Z) o longitud de onda.

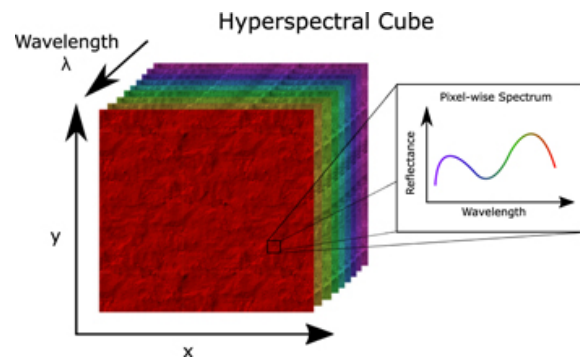


Fig. 1: Representación esquemática de un cubo de datos hiperespectral, mostrando las coordenadas espaciales y la firma espectral por píxel [13].

Esta resolución espectral permite identificar la composición química y física de los materiales topografiados con una gran precisión. Sin embargo, genera volúmenes masivos de información. El Comité Consultivo para Sistemas de Datos Espaciales (CCSDS) establece estándares internacionales rigurosos para la transmisión de este volumen de datos desde los satélites a las estaciones terrestres, priorizando algoritmos que maximicen el ancho de banda disponible [12].

En el ámbito del procesamiento de imágenes satelitales, la compresión se divide en dos grandes paradigmas: con pérdidas (*lossy*) y sin pérdidas (*lossless*). Dado el alto valor científico y observacional de los datos hiperespectrales, donde cualquier alteración del valor original del píxel puede corromper el análisis posterior (por ejemplo, en la detección de minerales o estudios climáticos), el uso de compresión sin pérdidas es un requisito estricto en la mayoría de las misiones [2].

Históricamente, herramientas de compresión de propósito general como GZIP (basado en el algoritmo Deflate, que combina LZ77 y codificación Huffman) han demostrado ser eficaces en la reducción absoluta del tamaño del archivo. No obstante, este enfoque de “caja negra” o compresión en bloque (*bulk compression*) presenta una limitación severa en entornos de análisis de datos de gran escala: la falta de acceso aleatorio (*random access*). Para poder consultar la firma espectral de un único píxel situado en las coordenadas (x, y, z), un sistema basado en compresores tradicionales obliga a la CPU a descomprimir secuencialmente la totalidad del fichero —o bloques masivos del mismo— en la memoria RAM.

Esta ineficiencia computacional, ha motivado la transición hacia las Estructuras de Datos Compactas (CDS)

[6], las cuales permiten que los datos permanezcan en su estado comprimido en la memoria mientras ofrecen soporte para consultas de navegación y extracción directa.

3.2. Estructuras de Datos Compactas y k^2 -raster

Las Estructuras de Datos Compactas (CDS) representan un cambio de paradigma frente a la compresión tradicional. Su objetivo principal es mantener la información en un formato comprimido directamente en la memoria principal, albergando en su propio diseño los mecanismos necesarios para soportar consultas de navegación y extracción de datos sin requerir fases previas de descompresión total [6].

Dentro de esta familia de estructuras, el k^2 -raster [1, 2] se erige como una solución optimizada para matrices bidimensionales de valores enteros, como es el caso de las imágenes hiperespectrales o los Modelos Digitales de Elevación (DEM). Su arquitectura se basa en una subdivisión espacial recursiva, conceptualmente análoga a un árbol cuaternario (quadtree), pero generalizada para dividir la matriz en $k \times k$ submatrices en cada nivel de profundidad.

El proceso de construcción del k^2 -raster sigue un enfoque jerárquico. Partiendo de la matriz original, se obtienen el valor máximo y el valor mínimo de toda la imagen, los cuales conforman la raíz del árbol. A continuación, la matriz se subdivide iterativamente. Para cada nueva submatriz, se calculan de nuevo sus respectivos máximos y mínimos locales. La recursividad se detiene cuando una submatriz alcanza un tamaño de 1×1 (un único píxel) o cuando todos los valores dentro de la submatriz son idénticos (es decir, el máximo local es igual al mínimo local), lo que permite podar ramas enteras del árbol y ahorrar un espacio de almacenamiento considerable.

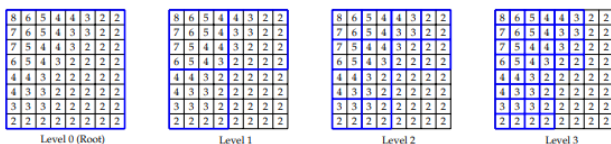


Fig. 2: Esquema conceptual de la partición espacial del k^2 -raster [3].

La clave de la alta tasa de compresión del k^2 -raster reside en su codificación diferencial predictiva [2]. En lugar de almacenar los valores absolutos extraídos en cada cuadrante (lo cual ocuparía el mismo espacio que la imagen original), la estructura almacena la diferencia matemática entre el valor del nodo padre y el de sus hijos. Dado que existe una alta autocorrelación espacial en las imágenes satelitales (los píxeles adyacentes suelen tener valores muy similares), esta resta produce secuencias de enteros positivos extremadamente pequeños.

Físicamente, la topología resultante del árbol se serializa en varios vectores unidimensionales:

- **Un mapa de bits (T):** Representa la estructura del árbol, indicando con un '1' si un nodo tiene hijos y con un '0' si es una hoja.

- **La matriz de máximos diferenciales (LMAX):** Contiene las diferencias entre el máximo de cada nodo padre y el máximo de sus hijos.
- **La matriz de mínimos diferenciales (LMIN):** Contiene las diferencias entre el mínimo de los hijos y el mínimo del padre.

Aunque la codificación diferencial reduce drásticamente la magnitud de los valores, la distribución estadística de la matriz LMAX presenta una particularidad crítica: está dominada por una inmensa mayoría de valores cercanos a cero, pero contiene una “larga cola” (*long tail*) de valores atípicos (*outliers*) de alta varianza, correspondientes a bordes afilados o anomalías en la imagen original [2].

Mientras que las regiones homogéneas de la imagen (como una extensión de agua o un bosque denso) generan diferencias cercanas a cero, las zonas de transición abrupta (como la costa costera, carreteras o edificaciones artificiales) introducen variaciones extremas en la firma espectral. Esta dualidad estadística exige que el codificador elegido sea lo suficientemente versátil para representar secuencias continuas de ceros, sin colapsar ante la aparición repentina de picos de alta entropía.

Codificar eficientemente esta distribución de enteros de forma compacta es el desafío principal que abordan las diferentes familias de codificadores analizadas a continuación.

3.3. Codificación de longitud variable y algoritmos entrópicos

Para comprimir las secuencias de diferencias generadas por la topología del k^2 -raster, se emplean Códigos de Longitud Variable (VLC, por sus siglas en inglés) [3]. Fundamentados en la teoría de la información de Shannon, estos algoritmos estadísticos abandonan la representación clásica de tamaño fijo (donde cada entero ocupa 32 o 64 bits) y asignan secuencias de bits cortas a los valores más frecuentes y secuencias más largas a los valores atípicos.

3.3.1. Códigos de Golomb y Rice

La codificación propuesta por Golomb [8] es una técnica parametrizada diseñada para distribuciones geométricas. Dado un entero $N \geq 0$ y un parámetro divisor $M > 0$, el algoritmo divide el número en dos partes: un cociente $q = \lfloor N/M \rfloor$ y un resto $r = N \bmod M$.

El código final se construye concatenando ambas partes: el cociente q se codifica en formato unario (representado por q bits a '1' seguidos de un '0' delimitador), mientras que el resto r se codifica en formato binario truncado.

La codificación de Rice es un caso particular y computacionalmente optimizado de Golomb, donde el parámetro divisor se restringe a potencias de dos ($M = 2^k$). Esta restricción matemática es fundamental a nivel de software, ya que permite sustituir las costosas instrucciones de división y módulo de la CPU por operaciones de desplazamiento de bits (*bit-shifts*) y máscaras lógicas ($\gg k$ y $\&((1 \ll k) - 1)$), acelerando drásticamente el tiempo de compresión y descompresión.

3.3.2. Códigos de Elias

A diferencia de Golomb y Rice, que requieren un barrido previo de los datos para calcular el parámetro M u óptimo, Peter Elias introdujo un conjunto de códigos universales [9] que no dependen de la distribución exacta de probabilidad de la fuente.

Elias Gamma (γ): Representa un entero $N > 0$ aislando su magnitud. Codifica la longitud de la representación binaria de N en formato unario, seguida de la propia representación binaria omitiendo el bit más significativo (el cual es siempre ‘1’ de forma implícita).

Elias Delta (δ): Diseñado para escalar mejor ante números grandes, aplica un enfoque recursivo. Codifica la longitud de la representación binaria utilizando Elias Gamma, seguida de los bits del valor propiamente dicho.

Elias-Fano: Desarrollado de forma independiente por Peter Elias y Robert Fano, este esquema divide la representación binaria de los enteros en “bits superiores” (*upper bits*) y “bits inferiores” (*lower bits*).

Los bits inferiores se almacenan de forma explícita y contigua, mientras que los bits superiores se codifican en unario, indicando el número de valores que comparten el mismo prefijo. Aunque es conceptualmente ideal para secuencias monótonas crecientes (como sumas parciales), su adaptación a secuencias de enteros generales permite una decodificación altamente veloz, posicionándose como un híbrido excelente entre compresión y latencia.

TABLA 1: COMPARATIVA DE CODIFICACIÓN

Algoritmo	Parámetro	N = 5	N = 14	N = 42
Golomb	$M = 4$	10 01	1110 10	1111111110 10
Rice	$k = 2$	10 01	1110 10	1111111110 10
Elias Gamma (γ)	N/A	001 01	0001 110	000001 01010
Elias Delta (δ)	N/A	011 01	00100 110	00110 01010
Elias-Fano*	N/A	N/A	N/A	N/A

*Nota: Elias-Fano requiere conocer el número total de elementos y el

valor máximo de una secuencia monótona creciente para calcular la distribución de bits superiores e inferiores, por lo que no es posible codificar enteros aislados fuera de su contexto.

3.3.3. Codificación de Huffman y el problema de la alta varianza

El algoritmo de Huffman [7] genera el código de prefijos óptimo para un alfabeto dado basándose en las frecuencias absolutas de aparición de cada símbolo. Mediante la construcción de un árbol binario estructurado *bottom-up*, los símbolos más probables se ubican cerca de la raíz, requiriendo menos bits para su representación.

Si bien es teóricamente óptimo para muchas distribuciones, presenta una limitación crítica ante matrices espaciales como LMAX. La inmensa varianza de los datos (*outliers* extremos) provoca una degeneración en la topología del árbol de Huffman, generando ramas de una profundidad desmesurada.

Esta asimetría matemática se traduce en códigos de prefijo que superan el tamaño de la palabra de máquina estándar (32 o 64 bits), suceso que se traduce en desbordamientos de enteros (*integer overflows*).

3.4. Codificadores alineados a palabra (*word-aligned*)

Como alternativa al procesamiento a nivel de bit característico de los algoritmos entrópicos, la literatura especializada propone el uso de codificadores alineados a palabra (*word-aligned*) [4]. Las arquitecturas de las unidades centrales de procesamiento (CPU) modernas están optimizadas para transferir y operar con bloques de memoria de tamaño fijo (típicamente de 32 o 64 bits). Los códigos de longitud variable, al cruzar los límites de estas palabras de memoria, incurrir en un coste computacional significativo debido a las operaciones de enmascaramiento y desplazamiento continuo.

Los enfoques alineados a palabra sacrifican una fracción marginal de la tasa de compresión absoluta a cambio de maximizar la velocidad de decodificación.

3.4.1. Familia Simple (Simple9 y Simple16)

El algoritmo Simple9, propuesto originalmente por Anh y Moffat [11], opera empaquetando el mayor número posible de enteros pequeños dentro de una única palabra de 32 bits. Para lograrlo, la palabra se divide rígidamente en dos secciones: un encabezado o selector de 4 bits y una sección de datos de 28 bits. El selector de 4 bits (que permite hasta 9 combinaciones válidas, de ahí su nombre) indica la configuración geométrica de los 28 bits restantes. Por ejemplo, el selector puede indicar que los datos albergan 28 enteros de 1 bit, 14 enteros de 2 bits, o 1 entero de 28 bits.

Simple16 es una evolución directa de este concepto que expande las combinaciones del selector a 16 esquemas posibles, aprovechando configuraciones asimétricas de empaquetado para reducir aún más el desperdicio de bits internos [11].

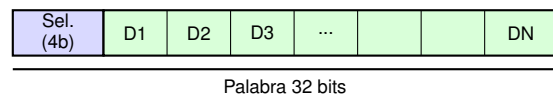


Fig. 3: Esquema conceptual de una palabra Simple9.

3.4.2. PForDelta (Patched Frame-of-Reference)

El algoritmo PForDelta [10] representa uno de los esquemas orientados a CPU más eficientes en la literatura de bases de datos y compresión. A diferencia de la familia Simple, PForDelta procesa la información en bloques fijos de gran tamaño (habitualmente en múltiplos de 32, 64 o 128 enteros).

Su mecánica principal consiste en determinar un ancho de bit fijo b que sea suficiente para representar la inmensa mayoría de los valores del bloque. Dado que la matriz LMAX del k^2 -raster contiene predominantemente valores muy pequeños, un b reducido es estadísticamente viable. Todos los valores del bloque se codifican forzosamente usando b bits, lo que permite a la CPU desenrollar bucles (*loop unrolling*) y leer los datos en menos ciclos de reloj.

Los valores que exceden la capacidad de b bits (denominados excepciones u *outliers*) se codifican truncando sus b bits inferiores en el bloque principal, mientras que sus bits superiores se almacenan en una lista de excepciones separada (el “parche” o *patch*). Para reconstruir la secuencia, el decodificador fusiona el flujo base con la lista de excepciones mediante aritmética de punteros simple. Este enfoque desplaza el coste del procesamiento de las excepciones al momento de la compresión, garantizando tiempos de descompresión extremadamente rápidos y consistentes [4].

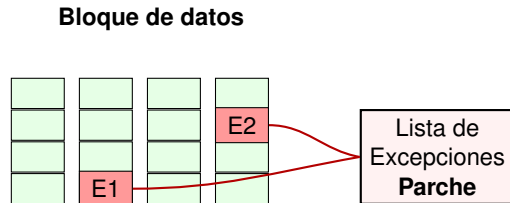


Fig. 4: Mecánica de PForDelta: Extracción de valores atípicos (*Excepciones*) hacia la estructura de parche.

3.5. Acceso aleatorio en la codificación secuencial

La principal debilidad de la Codificación de Longitud Variable (VLC) expuesta en la sección 3.3 es su secuencialidad. A diferencia de las estructuras tradicionales no comprimidas o los esquemas *word-aligned*, en un flujo de bits entrópico cada píxel codificado tiene una longitud impredecible.

En consecuencia, para recuperar el valor del i -ésimo elemento de una matriz comprimida mediante Golomb o Elias, un decodificador estándar está obligado a leer y decodificar todos los $i - 1$ elementos que le preceden para averiguar en qué bit exacto de la memoria comienza la secuencia del elemento deseado. Esta limitación destruye la viabilidad de la consulta de valores puntuales en grandes volúmenes de datos espaciales, obligando al sistema a operar bajo un paradigma de descompresión en bloque (*bulk decompression*).

Para dotar a la codificación de longitud variable de verdaderas capacidades de acceso aleatorio, es necesario romper esta dependencia secuencial. Este objetivo se logra mediante la introducción de un balanceo entre el tiempo de acceso y el espacio de almacenamiento temporal, implementando un mecanismo de sincronización basado en hitos o punteros de salto (*milestones*).

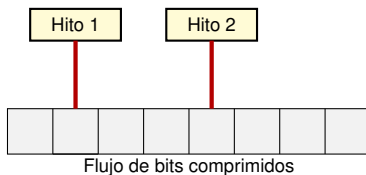


Fig. 5: Acceso aleatorio: Tabla de hitos apuntando a bloques en memoria.

El mecanismo opera definiendo una tasa de muestreo constante S (por ejemplo, $S = 64$ o $S = 128$ elemen-

tos). Durante la fase de compresión, además de generar el flujo de bits continuos, el codificador almacena el desplazamiento absoluto (*bit-offset*) del primer bit de cada S -ésimo elemento en una estructura auxiliar en memoria RAM (vector de posiciones).

Durante la fase de consulta, si el sistema requiere extraer el elemento situado en el índice i , la CPU calcula matemáticamente el hito más cercano mediante la división entera $m = \lfloor i/S \rfloor \cdot A$. A continuación, el decodificador realiza un salto de memoria directo hacia el *bit-offset* almacenado en el índice m del vector auxiliar, y desde esa posición, decodifica secuencialmente únicamente los residuos necesarios ($i \bmod S$) hasta alcanzar el elemento objetivo.

Este enfoque garantiza un límite superior para la latencia de decodificación (nunca se decodifican más de $S - 1$ elementos), transformando un problema de coste lineal $\mathcal{O}(N)$ en uno de coste constante $\mathcal{O}(S)$ acotado. El diseño, implementación a bajo nivel y acoplamiento de este mecanismo de hitos para las diversas familias de codificadores entrópicos dentro del motor del k^2 -raster constituye el núcleo metodológico de este trabajo.

4 METODOLOGÍA Y DESARROLLO

El desarrollo metodológico de este proyecto se estructuró sobre una base algorítmica preexistente en lenguaje C, la cual implementaba la partición espacial del k^2 -raster y los codificadores de referencia (DACS, Simple9, Simple16 y PForDelta). A partir de esta infraestructura, el trabajo se centró en la integración nativa de nuevas familias entrópicas y en la validación automatizada de su rendimiento.

4.1. Integración de codificadores entrópicos

El trabajo partió de la base de código original implementada en C. Se desarrollaron módulos algorítmicos independientes para cada una de las familias de codificación entrópica analizadas en el estado del arte: Golomb, Rice, la familia Elias (Gamma, Delta y Fano) y Huffman.

Cada módulo se diseñó con dos rutinas principales interdependientes: una función de codificación que transforma las secuencias de diferencias (focalizadas en las matrices LMAX y LMIN) en un flujo de bits comprimido de longitud variable, y una función de decodificación encargada de recuperar el valor original a partir del flujo comprimido.

4.2. Acceso aleatorio mediante hitos

El motor original del k^2 -raster ya disponía de un mecanismo para soportar el acceso aleatorio, basado en una tasa de muestreo constante (SAMPLING-FREQUENCY). Este sistema registra un puntero de salto (*milestone*) en una estructura auxiliar (*array_pos*) cada 30 elementos codificados.

El principal reto consistió en acoplar la naturaleza secuencial de los nuevos codificadores entrópicos a esta arquitectura rígida de hitos. Se adaptaron las lógicas internas

de Golomb, Rice, Elias y Huffman para que, de forma paralela a la compresión, determinaran y registraran su desplazamiento absoluto a nivel de bit (*bit-offset*).

De este modo, al recibir una consulta espacial, el decodificador es capaz de calcular el hito más cercano mediante una división entera, saltar a la dirección de memoria exacta y decodificar exclusivamente el residuo necesario, logrando una integración transparente con el motor principal.

4.3. Entorno de validación y extracción de métricas

Dado el carácter estricto de compresión sin pérdidas (*lossless*) requerido en imágenes satelitales, fue necesario garantizar que los nuevos algoritmos no corrompieran la información espectral original. Para ello, se integraron los diferentes algoritmos implementados a las diferentes funciones de verificación existentes en el código (`verify_saved_data_simple` y `test_get_cell`) y se crearon diferentes archivos para probar ejemplos en los algoritmos y comprobar su correcto funcionamiento.

Estas funciones simulan consultas aleatorias masivas y realizan una comparación determinista (bit a bit) entre la imagen en bruto y la matriz decodificada en tiempo de ejecución, asegurando la integridad estructural de los datos. Dado que la estructura k^2 -raster se basa en la codificación diferencial jerárquica, la extracción de un solo píxel obliga al decodificador a recorrer la topología del árbol desde la raíz (valores globales) hasta las hojas, resolviendo las diferencias almacenadas en LMAX y LMIN. Por tanto, la validación de píxeles garantiza que toda la cadena de nodos intermedios se mantiene consistente tras la compresión entrópica, ya que cualquier error de desplazamiento de bits en los niveles superiores corrompería el valor final.

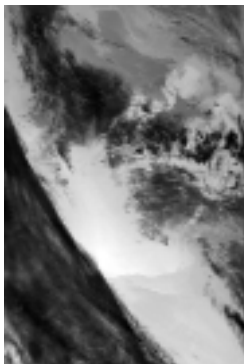


Fig. 6: Visualización de la escena hiperespectral AIRS utilizada en las pruebas de validación.

Una vez garantizada la fiabilidad, se desarrolló un entorno automatizado en Python que ejecuta cada prueba sobre 100.000 iteraciones para asegurar significancia estadística. Las métricas capturadas incluyen:

- **Eficiencia Temporal (Latencia):** Extracción del reloj nativo de C (`gettimeofday()`) en microsegundos.
- **Eficiencia Espacial (RAM):** Monitorización del pico dinámico mediante `/usr/bin/time -v`.

- **Tasa de Compresión (Bitrate):** Cálculo analítico en bits por píxel (bpp).

5 EXPOSICIÓN DE RESULTADOS

En esta sección se discuten los resultados obtenidos. El análisis se articula en torno a la eficiencia espacial, los límites arquitectónicos de los algoritmos y la eficiencia temporal por consulta, consolidando los datos bajo variaciones del parámetro de partición k .

5.1. Eficiencia espacial y distribución estructural

Al desglosar internamente el peso de las estructuras comprimidas, se evidencia que la matriz diferencial de máximos (LMAX) constituye el núcleo del peso del fichero, acaparrando típicamente más del 90 % del espacio.

La Tabla 2 consolida el impacto del parámetro de partición k sobre la tasa de compresión final en los distintos algoritmos. Los resultados demuestran que $k = 6$ representa un punto de equilibrio óptimo, minimizando la huella de memoria sin sobrecargar la topología de los nodos intermedios.

5.2. Límites arquitectónicos: El colapso de Huffman

Si bien la teoría de la información posiciona a la codificación de Huffman como el código de prefijos óptimo, su aplicación directa sobre la matriz LMAX evidenció una viabilidad nula en la práctica, escalando hasta los 67.8 MB de peso.

En imágenes satelitales, las zonas de transición (bordes) generan diferencias de alta magnitud. El algoritmo de Huffman asigna cadenas de bits extremadamente largas a estos valores atípicos (*outliers*). En nuestra implementación nativa, cuando la longitud del código asignado excede la capacidad de representación de la palabra de máquina (32 bits), se producen desbordamientos de enteros (*integer overflows*) en las variables `uint32_t`, corrompiendo irrecuperablemente la imagen. Esto demuestra que los codificadores de frecuencia estricta son incompatibles con distribuciones de alta entropía no mitigada.

5.3. Eficiencia temporal: Latencia de consulta

Para evaluar la viabilidad en sistemas interactivos, se midió el tiempo necesario para aislar un único píxel. La Tabla 3 resume los resultados, confirmando el dominio de la familia alineada a palabra (*word-aligned*). PForDelta logra latencias prácticamente nulas ($0.09 \mu s$ con $k = 6$), muy inferiores al algoritmo base DACS.

Por otro lado, los codificadores entrópicos integrados (Golomb y Rice) registran latencias sumamente competitivas, en torno a los $2 \mu s$. Este éxito certifica la eficacia metodológica de la integración del sistema de hitos (`array-pos`). Es importante destacar que esta latencia por

TABLA 2: RESUMEN COMPARATIVO DE EFICIENCIA ESPACIAL VARIANDO EL PARÁMETRO DE SUBDIVISIÓN k .

Algoritmo	Total (MB)			LMAX (MB)			Bitrate (bpp)		
	$k = 2$	$k = 6$	$k = 12$	$k = 2$	$k = 6$	$k = 12$	$k = 2$	$k = 6$	$k = 12$
Gzip (Baseline)	22.09	22.09	22.09	N/A	N/A	N/A	10.16	10.16	10.16
DACS (Nativo)	28.39	20.63	24.41	19.25	19.58	24.07	13.06	9.49	11.23
PForDelta_64	33.71	20.84	23.84	23.23	19.81	23.52	15.51	9.59	10.97
Simple16	30.95	21.27	25.98	21.15	20.26	25.69	14.24	9.78	11.95
Simple9	32.24	21.86	26.36	22.21	20.84	26.06	14.83	10.06	12.13
Golomb	35.73	23.69	28.30	24.89	22.65	28.02	16.44	10.90	13.02
Rice	36.75	24.11	28.71	25.66	23.06	28.43	16.90	11.09	13.21
Elias Gamma	34.11	29.93	37.63	23.81	28.66	37.26	15.69	13.77	17.31
Elias Delta	33.13	27.79	33.71	23.15	26.66	33.40	15.24	12.78	15.50
Elias Fano	41.39	27.33	32.15	29.39	26.19	31.84	19.04	12.57	14.79
Huffman	78.12	67.87	65.98	44.95	53.76	62.18	35.94	31.22	30.35

TABLA 3: RESUMEN COMPARATIVO DE RENDIMIENTO TEMPORAL: LATENCIA MEDIA DE CONSULTA Y CONSUMO DINÁMICO DE RAM.

Algoritmo	Latencia (μs)			RAM (MB)		
	$k = 2$	$k = 6$	$k = 12$	$k = 2$	$k = 6$	$k = 12$
PForDelta_64	0.058	0.090	0.119	96.19	75.55	81.84
Simple16	0.641	0.380	0.436	1.79	1.58	1.81
DACS (Nativo)	2.077	0.954	1.022	83.38	80.64	89.21
Simple9	1.896	1.468	1.633	1.62	1.62	1.73
Rice	3.731	1.663	1.243	70.05	63.62	68.87
Golomb	3.761	2.315	1.261	69.34	63.29	68.16
Elias Gamma	3.462	1.721	1.392	68.12	69.17	77.60
Elias Delta	3.251	1.620	1.333	67.82	67.29	73.71
Elias Fano	3.640	1.710	1.304	73.86	66.85	72.11
Huffman	5.076	3.100	2.356	64.88	61.39	64.24

consulta incluye no solo el tiempo intrínseco de decodificación, sino también el coste de resolución espacial del árbol cuaternario.

Cada vez que se solicita un píxel, el sistema debe descender desde la matriz raíz, navegando por el mapa de bits (T) e interpretando los residuos en LMAX y LMIN hasta alcanzar la hoja correspondiente. El hecho de que Golomb y Rice mantengan tiempos de respuesta tan consistentes demuestra que la sobrecarga introducida por el procesamiento de bits de longitud variable queda enmascarada y absorbida por la jerarquía predictiva del propio k^2 -raster.

5.4. Impacto de la topología en el rendimiento global

La evaluación revela que el parámetro k actúa como el factor de modulación principal. Un valor bajo ($k = 2$) profundiza el árbol, forzando al decodificador a recorrer múltiples niveles, lo que empeora las latencias generales. Sin embargo, un valor demasiado alto ($k = 12$) aplanan excesivamente la estructura, perjudicando la capacidad de predecir diferencias locales y deteriorando el *bitrate*.

Asimismo, se observa un claro compromiso espacial-temporal, destacando distintos perfiles según el algoritmo. PForDelta_64 se consolida como la opción idónea cuando la métrica crítica es la velocidad de CPU (latencia casi nula de $0.090 \mu s$), logrando además una compresión (9.59 bpp) muy cercana al estándar nativo DACS (9.49 bpp), aunque

a costa de mantener picos altos de RAM dinámica (75.55 MB). Por el contrario, la familia *word-aligned* (y en particular Simple16) emerge como la solución más eficiente para sistemas con memoria restringida: mantiene latencias sub-microsegundo ($0.380 \mu s$) y una compresión altamente competitiva (9.78 bpp), pero reduciendo drásticamente el consumo dinámico de RAM a menos de 2 MB, superando ampliamente en eficiencia global a los codificadores entrópicos clásicos.

6 CONCLUSIONES

El presente Trabajo de Fin de Grado ha cumplido con éxito su objetivo principal al lograr la integración nativa y operativa de diversas familias de codificación entrópica sobre la base arquitectónica del k^2 -raster para la compresión sin pérdidas de imágenes hiperespectrales.

Uno de los aportes metodológicos más relevantes ha sido la validación del sistema de sincronización mediante hitos. Esta solución ha demostrado ser efectiva para romper la dependencia secuencial de los Códigos de Longitud Variable (VLC). Al acoplar el registro de posiciones absolutas (*array_pos*) a la lógica de los codificadores de bits, se ha logrado dotar a la estructura de un acceso aleatorio, posibilitando la extracción directa de píxeles individuales con una latencia controlada y compatible con sistemas de consulta en tiempo real.

A nivel analítico, la investigación ha puesto de manifies-

to las severas limitaciones estructurales de los diccionarios clásicos de frecuencia estricta. El colapso de la codificación de Huffman ratifica que las secuencias de diferencias con alta entropía son incompatibles con arquitecturas limitadas por el tamaño de la palabra de máquina estándar.

Finalmente, el análisis de las métricas concluye que los enfoques alineados a palabra (*word-aligned*) dominan el espacio de soluciones prácticas para esta estructura. Para misiones espaciales donde la métrica crítica sea la latencia de procesamiento, PForDelta_64 resulta imbatible. Por otro lado, para escenarios con restricciones severas de memoria, la familia Simple (destacando Simple16) se posiciona como la alternativa óptima, reduciendo el consumo dinámico de RAM a menos de 2 MB sin sacrificar competitividad en la tasa de compresión. Por su parte, aunque las integraciones de Golomb y Rice han servido para validar con éxito la viabilidad teórica y funcional del acceso aleatorio en flujos de bits, su mayor huella de memoria los relega a un segundo plano frente a la eficiencia global de los métodos alineados a palabra en este contexto específico.

USO DE LA IA GENERATIVA

En la realización de este Trabajo de Fin de Grado, he utilizado el modelo de Inteligencia Artificial generativa Gemini (desarrollado por Google) principalmente como apoyo en cuatro frentes:

- **Soporte teórico:** Me ha servido de gran ayuda para entender mejor los algoritmos más abstractos. Cuando me surgían dudas teóricas, lo he utilizado para solicitar ejemplos prácticos y visuales que me ayudaran a aclarar los conceptos de forma más sencilla.
- **Soporte en la programación:** Lo he usado como asistente para resolver dudas puntuales y lograr salir adelante cuando me topaba con errores o problemas específicos al escribir el código en C.
- **Generación de gráficos:** Me ha servido como herramienta de asistencia para generar y afinar el código LaTeX (basado en la librería TikZ) perteneciente a las Figuras 3, 4 y 5 de este documento.
- **Revisión de textos:** Finalmente, me ha sido útil para revisar la ortografía y optimizar el estilo de redacción tanto de los informes de progreso que he ido entregando como de esta memoria final.

AGRADECIMIENTOS

Quisiera expresar mi más sincero agradecimiento a mi tutor, Hing Fai Kevin Chow, por su guía, paciencia y apoyo constante durante todo el desarrollo de este proyecto. Sus conocimientos técnicos y su base de investigación han sido un pilar fundamental para llevar a buen puerto este trabajo. Asimismo, agradezco a mi familia y amigos por su apoyo incondicional y ánimos durante todos mis años de formación universitaria.

REFERENCIAS

- [1] H. F. K. Chow, D. E. O. Tzamarias, I. Blanes Garcia, y J. Serra-Sagrista, "Introducing k^3 -raster compact data structure for hyperspectral images", en *6th International Workshop on On-Board Payload Data Compression (OBPDC)*, sept. 2018.
- [2] K. Chow, D. E. O. Tzamarias, I. Blanes, y J. Serra-Sagristà, "Using predictive and differential methods with k^2 -raster compact data structure for hyperspectral image lossless compression", *Remote Sensing*, vol. 11, no. 21, p. 2461, 2019.
- [3] K. Chow, D. E. O. Tzamarias, M. Hernández-Cabronero, I. Blanes, y J. Serra-Sagristà, "Analysis of variable-length codes for integer encoding in hyperspectral data compression with the k^2 -raster compact data structure", *Remote Sensing*, vol. 12, no. 12, p. 1983, 2020.
- [4] K. Chow, D. E. O. Tzamarias, M. Hernández-Cabronero, I. Blanes, y J. Serra-Sagrista, "Performance improvement on k^2 -raster compact data structure for hyperspectral scenes", *IEEE Geoscience and Remote Sensing Letters*, vol. 19, pp. 1-5, 2021.
- [5] Universidad de Concepción, "Bitte ein Bit: Encoding & Decoding Wiki". [En línea]. Disponible en: <https://edaa.inf.udec.cl/Encoding/>. Consultado: 20 de mayo, 2026.
- [6] G. Navarro, *Compact Data Structures: A Practical Approach*, Cambridge University Press, 2016.
- [7] D. A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes", *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098-1101, Sept. 1952.
- [8] S. W. Golomb, "Run-length encodings", *IEEE Transactions on Information Theory*, vol. 12, no. 3, pp. 399-401, Jul. 1966.
- [9] P. Elias, "Universal codeword sets and representations of the integers", *IEEE Transactions on Information Theory*, vol. 21, no. 2, pp. 194-203, Mar. 1975.
- [10] M. Zukowski, S. Heman, N. Nes, y P. Boncz, "Super-Scalar RAM-CPU Cache Compression", en *Proceedings of the 22nd International Conference on Data Engineering (ICDE)*, 2006, pp. 59-59.
- [11] A. Anh y A. Moffat, "Index compression using 64-bit words", *Software: Practice and Experience*, vol. 40, no. 2, pp. 131-147, 2010.
- [12] Consultative Committee for Space Data Systems (CCSDS), *Lossless Multispectral & Hyperspectral Image Compression*, CCSDS 123.0-B-2, Blue Book, Washington, DC, USA, 2019.
- [13] T. W. Sawyer, A. S. Luthman, y S. E. Bohndiek, "Evaluation of illumination system uniformity for wide-field biomedical hyperspectral imaging", *Journal of Optics*, vol. 19, no. 4, 045301, 2017.

APÉNDICE

A.1. Diagrama de Gantt de la planificación inicial

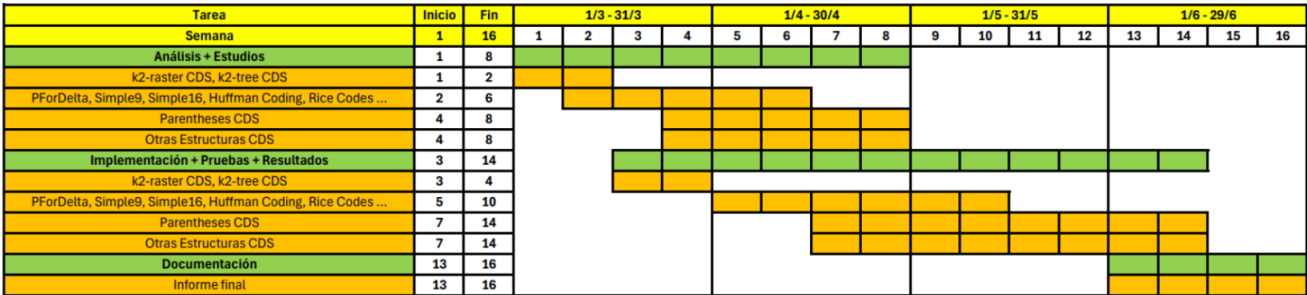


Fig. 7: Diagrama de Gantt perteneciente a la planificación inicial de este trabajo.