Compresión de imágenes de satélite a alta velocidad

Carlos Vioque Moreno

Resumen– En este artículo queda reflejado todo el proceso de desarrollo de optimización del estándar CCSDS 123.0-B-1. El proyecto parte de una modificación anterior que implementa un codificador aritmético contextual sustituyendo el codificador entrópico original. Se trabaja sobre la base teórica de que imágenes multiespecrates e hiperespectrales, tomadas por el mismo sensor de observación terrestre, tienden a compartir similitudes estadísticas. La propuesta de optimización se basa en la implementación de una look up table, generada por una o varias imágenes del mismo sensor, para sustituir el proceso del cálculo del contexto de cada bit que realiza el codificador aritmético contextual. También se ha estudiado el impacto que supone la selección de una look up table correcta y una errónea.

Palabras clave– imagen, imagen multiespectral, imagen hipersespectral, compresión, look up table, CCSDS, CCSDS-123, satélite, optimización, sensor, observación terrestre

Abstract– This article reflects the process of developing an optimization of CCSDS123.0-B-1 standard. The project starts from a previous modification which implements a contextual arithmetic coder in replacement of the original entropy encoder. It works on the teoric basis that multispectral and hyperspectral images, normally, share statistic similarities. The optimization proposal is an implementation of a look up table, generated by one or multiple images from the same device, in replacement of the context calculation process, that the contextual arithmetic coder does for every bit. Also it has been tested the difference between a good look up table or a bad one.

Keywords– image, multispectral image, hyperspectral image, compression, look up table, CCSDS, CCSDS-123, satellite, optimization, sensor, satellite imagery

1 Introducción - Contexto del trabajo

Actualmente hay, aproximadamente, unos 1900 satélites artificiales orbitando la Tierra [1], de los cuales una parte están dedicados a la observación terrestre. Estos satélites toman imágenes de la superficie terrestre, las comprimen y las envían a las estaciones terrestres. Este trabajo de final de grado (TFG) se ha basado en la optimización del estándar recomendado por el Consultative Committee for Space Data Systems (CCSDS) [2] para la compresión de imágenes multiespectrales e hiperespectrales para la observación terrestre: el CCSDS 123.0-B-1 [3], CCSDS-123 a partir de ahora. No se ha trabajado sobre la versión original sino que se hace con una versión que sustituye el codificador por entropía original por un codificador aritmético contextual. Esta modificación del CCSDS-123 se introduce en [4].

La propuesta de este TFG es sustituir el proceso mediante el cual se determina el contexto y la probabilidad para cada bit a codificar por una Look Up Table (LUT), con el objetivo de reducir la carga computacional evitando tener un gran impacto en el tamaño de los ficheros comprimidos.

En este artículo se hará una breve presentación de las imágenes multiespectrales e hiperespectrales, se justificará la propuesta y los cambios realizados, posteriormente se estudiarán los resultados de los experimentos y finalmente se propondrán varias vías de continuación.

2 Imágenes RGB, Multiespectrales e Hiperespectrales

El estándar CCSDS-123 está recomendado para trabajar con imágenes multiespecrtales e hiperespectrales. Una imagen digital RGB tiene 3 canales o bandas: rojo (R), verde (G) y azul (B). Este estándar especifica que cada canal tiene 8 bits, por lo tanto puede tomar valores desde 0 a 255,

• E-mail de contacte: carlosvioquem@gmail.com
• Mención realitzada: Tecnologies de la Informació
• Treball tutoritzat per: Joan Bartrina Rapesta (Enginyeria de la Informació i les comunicacions)
• Curs 2020/21

ambos incluidos.

Una imagen multiespectral está formada por más de 3 bandas y normalmente no supera las 20. Estas bandas están repartidas por el espectro electromagnético, no necesariamente con una distribuición proporcional ni contigua. Las bandas de una imagen multiespectral pueden estar tanto en el rango de luz visible (entre los 700nm y los 400nm aproximadamente (Fig. 1)) o estar en los rangos no visibles para el ojo humano, ultravioleta e infrarrojos, de esta forma las imágenes multiespectrales permiten extraer información adicional que el ojo humano no es capaz de capturar.

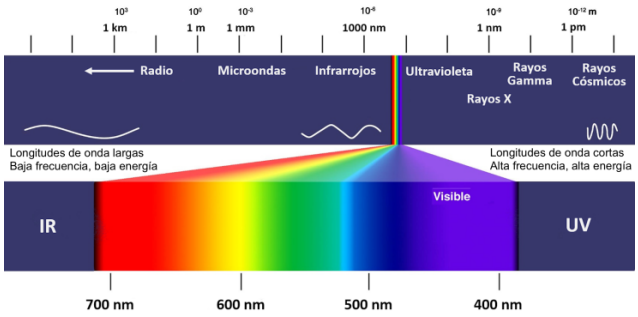

Fig. 1: Espectro electromagnético.

Las imágenes hiperespectrales están formadas por un número mucho mayor de bandas y estas son contiguas. Con una imagen multiespectral obtenemos los valores de intensidad en puntos concretos del espectro que capte la cámara, en cambio con una imagen hiperespectral obtenemos el espectro continuo [5].

3 Estado del arte

El CCSDS-123 está definido en 3 etapas. En la primera se obtiene la predicción p a partir de los datos a codificar x, con estos valores se calculan los residuos mediante $\Lambda = x - p$. A continuación se mapean los valores de Λ a enteros positivos para obtener λ. Finalmente el codificador por entropía codifica estos valores λ obteniendo el archivo comprimido, al que llamaremos $codestream$. En la Fig. 2 se muestra el esquema de esta codificación.

Fig. 2: Esquema del proceso de codificación de una imagen del estándar CCSDS-123

En este TFG se ha trabajado a partir de una modificación [4] que sustituye el codificador por entropía del CCSDS-123 por un codificador aritmético contextual. Este codificador procesa los datos λ banda a banda y línea a línea, a estos valores se les aplica una transformación para convertir la matriz 2D correspondiente a cada banda a una matriz 3D que contiene la representación binaria de los mismos valores. Podemos ver una representación gráfica en la Fig. 3

Son estos 0s y 1s sobre los cuales el codificador aritmético contextual calcula la probabilidad de que sean 0. Cuanto más certero sea este cálculo de la probabilidad, mejor compresión obtendremos. Para ello es necesario un proceso que determine el contexto para cada bit a codificar.

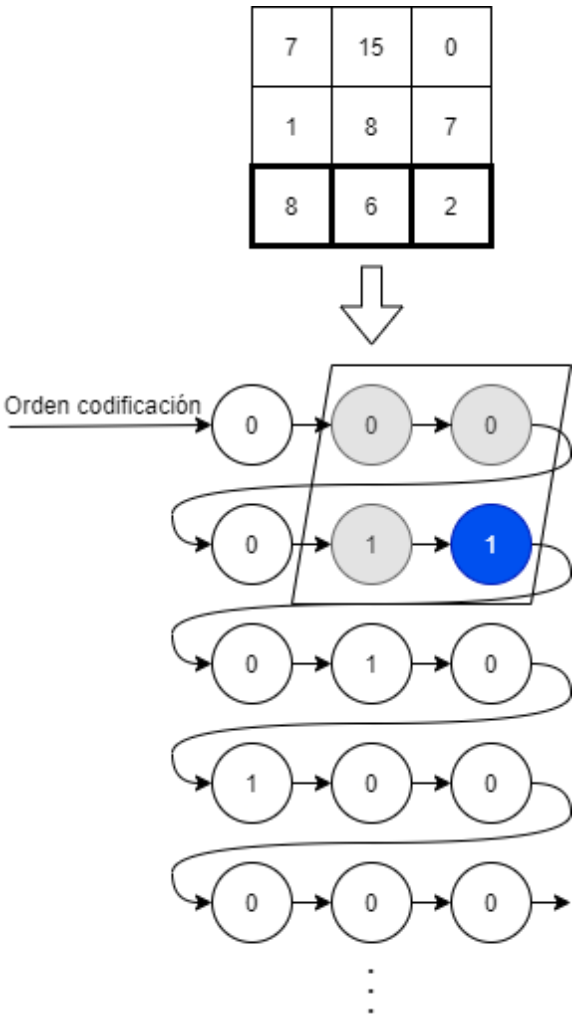

Fig. 3: Representación binaria de los valores λ procesados por el codificador y flujo de codificación.

3.1. Modelo contextual

Cada bit a codificar tiene una ventana contextual, en la Fig. 3 podemos ver en azul el bit a codificar y un recuadro con su ventana contextual, los vecinos están sombreados en gris. Supongamos que **M** es el conjunto de todos los posibles patrones que puede haber dentro esta ventana contextual, siendo m un caso concreto que pertenece a **M**, esto se traduce a un índice c que pertenece a **C**, por lo tanto **C** es el conjunto de índices correspondientes a **M**. El objetivo es determinar la probabilidad p de que cada bit sea 0 a partir de un contexto c:

$$p(b = 0|c)$$

Para determinar este contexto es necesaria una función F que a partir de m calcule c: $F(m) = c$. Esta función F utiliza los estados de significancia de los vecinos del bit b a codificar, es decir los bits que no son b que están dentro de la ventana contextual de b.

Por lo tanto, buscar el contexto para cada bit y calcular su probabilidad es muy costoso, por ello en este artículo se propone la sustitución de este proceso por una LUT que

aportará las probabilidades de que cada bit sea 0. Esta modificación se basa en la teoría en que diferentes imágenes tomadas por el mismo dispositivo tienden a compartir similitudes en sus estadísticas, por lo que podemos definir una LUT con una imagen y utilizar esa LUT para comprimir otras imágenes del mismo sensor.

4 MODIFICACIONES REALIZADAS

4.1. Look up table

Una LUT es una estructura de datos que se utiliza para sustituir un proceso computacional muy exigente. En este caso sustituimos el proceso de calcular el contexto para cada bit por una LUT que contiene la probabilidad de que un bit sea 0, basándose en el valor sus vecinos y en el bitplane que nos encontramos. Se utiliza una LUT tridimensional, la primera dimensión representa el valor residual del bit izquierdo, la segunda dimensión representa el valor residual del bit superior y la tercera dimensión representa el plano de bits en el que nos encontramos. Por lo tanto la probabilidad de que un bit sea 0 es:

$$Prob(bitplane_{i,j} = 0) = LUT[X_{i-1,j}][X_{i,j-1}][bitplane]$$

En la Fig. 4 podemos ver una representació gráfica del funcionamiento de la fórmula anterior.

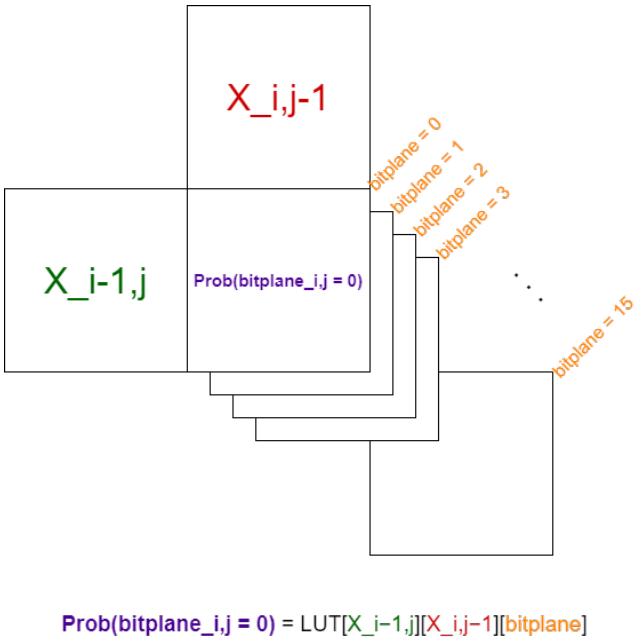

Fig. 4: Esquema de la consulta de la probabilidad en la LUT

Se ha implementado una LUT con dimensiones $LUT[256][256][16]$. Tenemos 16 bitplanes debido a que las imágenes con las que estamos trabajando son 16-bit Signed, por lo tanto cada $sample$ de cada bitplane puede tomar un valor entre 0 y 32.767. Si cada $sample$ puede tomar un valor entre 0 y 32.767, ¿Por qué las 2 primeras coordenadas de la LUT son 256 y no son ambas 32768? Una LUT como la que se propone, de dimensiones 256 x 256 x 16, tendría un tamaño en memoria de:

$$256 \cdot 256 \cdot 16 \cdot 32\, bits = 4MB$$

(32 bits por que estamos guardando los valores en variables tipo int). En cambio una LUT que contemple todos los casos posibles tendría un tamaño en memoria de:

$$32768 \cdot 32768 \cdot 16 \cdot 32\, bits = 64GB$$

Tomamos una imagen del conjunto de pruebas al azar. Se trata de una imagen de un tamaño de 512 filas · 680 columnas · 224 bandas. Esto quiere decir que tenemos un total de $512 \cdot 680 \cdot 224 = 77{,}987{,}840\ samples$, si se realiza una ejecución de la generación de la LUT a partir de esta imagen y se extrae cuantos valores residuales superiores a 256 hay, se obtiene un total de 374.180, es decir, los valores residuales superiores a 256 son el 0,48 %. Este error se asume para poder tener una LUT de 4MB, mucho más manejable que una LUT de 64GB. En los casos en los cuales los valores sean superiores a 256 se codificara el bit $bitplane_{i,j}$ considerando:

$$Prob(bitplane_{i,j} = 0) = 0,5$$

Además de estos casos problemáticos también existen conflictos al codificar los valores de la primera fila y la primera columna de cada banda, ya que en estos casos no pueden consultarse los residuales superior o derecho. En estos casos también se asume el error, que se trata del 0,34 % y se codificará igualmente considerando:

$$Prob(bitplane_{i,j} = 0) = 0,5$$

4.2. Optimización del proceso de codificación

El proceso original estaba configurado para ejecutarse por threads y que cada thread utilizase una instancia de la función codificadora $ArithmeticCoderFLW$ [6] que genera los bitstreams. Posteriormente el decodificador lee estos bitstreams y los asigna a cada thread. La modificación que ha realizado crea una instancia de la función codificadora por cada bitplane, es decir, pasamos de tener una instancia en cada thread a tener dieciséis por cada thread. Esta optimización se realizó para mejorar el rendimiento de la codificación, buscando que cada codificador aritmético codificase siempre probabilidades similares.

5 EXPERIMENTOS

5.1. Imágenes de teledetección

Inicialmente trabajamos con 2 conjuntos de imágenes de teledetección. 5 imágenes **hiperespectrales** del sensor AVIRIS [7], con 224 bandas y un tamaño de 680 de alto x 512 de ancho. Estas imágenes tienen la siguiente nomenclatura: $aviris\_yellowstone\_f060925t01p00r12\_$**scXX**$\_uncal\_224\_512\_680\_2\_0\_16\_0\_0\_0.png$ Las diferenciaremos por el código en negrita, sustituyendo scXX por sc00, sc03, sc10, sc 11, sc18. Podemos ver estas imágenes en la Tabla 1.

3 imágenes **multiespectrales** del sensor Landsat [8], con 6 bandas y un tamaño de 1024 x 1024. Estas imágenes tienen la siguiente nomenclatura: $Landsat\_$**xxxxxx**$\_6\_1024\_1024\_2\_0\_16\_0\_0\_0$ Las diferenciaremos entre sí por el código en negrita, sustituyendo xxxxxx por agriculture, coast y mountain. Podemos ver estas imágenes en la Tabla 2.

TABLA 1: IMÁGENES DEL SENSOR AVIRIS

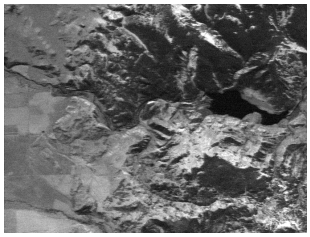

Imagen **sc00**

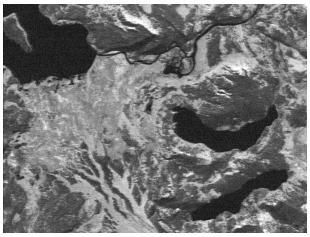

Imagen **sc03**

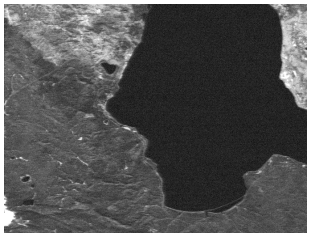

Imagen **sc10**

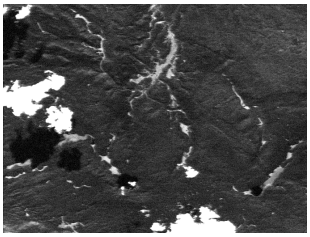

Imagen **sc11**

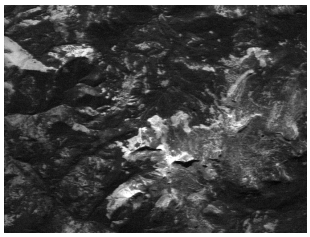

Imagen **sc18**

TABLA 2: IMÁGENES DEL SENSOR LANDSAT

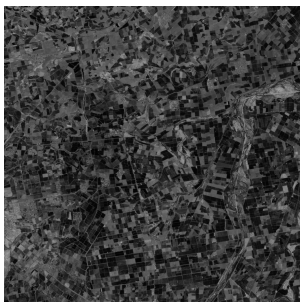

Imagen **agriculture**

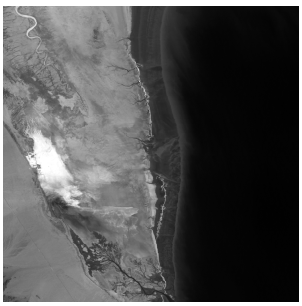

Imagen **coast**

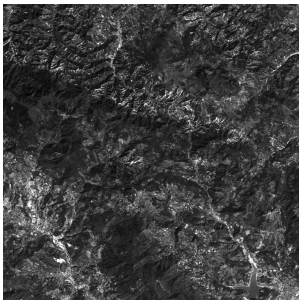

Imagen **mountain**

TABLA 3: RESULTADOS DE LAS 2 BATERÍAS DE EJECUCIONES SIN LUT

Sin LUT (Código Original)			
	Imagen sc03 (AVIRIS)		
Threads	*bps*	T. Cod. (ms)	T. Dec. (ms)
1	6,03	37780	42779
2	6,03	29083	32039
4	6,04	23512	25895
8	6,06	21439	22221
16	6,11	20379	20997
32	6,17	20134	21203
64	6,23	21714	22891
	Imagen agriculture (Landsat)		
Threads	*bps*	T. Cod. (ms)	T. Dec. (ms)
1	3,55	2897	3106
2	3,55	2356	2277
4	3,58	1947	1956
8	3,63	1965	1962
16	3,72	2068	2003
32	3,83	1944	2454
64	3,94	2177	2129

5.2. Optimización de la ejecución

El primer paso antes de iniciar las pruebas con las diferentes imágenes fué decidir con qué cantidad de threads se obtienen las ejecuciones más óptimas, para ello utilizamos la imagen **sc03** de AVIRIS y la imagen **agriculture** de Landsat. En total se hicieron 4 baterías de ejecuciones, 2 con cada imagen con y sin LUT. En cada bateria se ejecutó el proceso con 1, 2, 4, 8, 16, 32 y 64 threads, los resultados obtenidos los comparamos con la ejecución con un único thread para cuantificar las mejoras.

En la Tabla 3 tenemos los resultados de las ejecuciones con el código original y en la Tabla 4 están los resultados de las ejecuciones una vez implementada la LUT. Podemos ver que en todos los casos a partir de 8 threads la mejora de tiempo se estabiliza y no obtenemos prácticamente ninguna mejora, en cambio los bits por sample (*bps*) a partir de los 8 threads crecen de forma más acelerada. Este aumento en los *bps* es más importante en las ejecuciones con LUT. Si miramos el aumento de los *bps* porcentualmente en las ejecuciones con LUT (Tabla 5) podemos ver que cada aumento de threads provoca que el porcentaje que aumentan los *bps* sea exponencial. Esto se debe a lo expuesto en el apartado 4.1. Las primeras filas y columnas de cada división que crean los threads se codifican automáticamente con $Prob(bitplane_{i,j} = 0) = 0,5$, por lo tanto este comportamiento es lógico ya que estamos doblando el número de primeras columnas en cada ejecución, asumiendo estos errores.

Teniendo en cuenta todos estos aspectos se decidió realizar todo el resto de ejecuciones con 8 threads.

TABLA 4: RESULTADOS DE LAS 2 BATERÍAS DE EJECUCIONES CON LUT

Con LUT			
	Imagen sc03 (AVIRIS)		
Threads	*bps*	T. Cod. (ms)	T. Dec. (ms)
1	6,18	24158	32744
2	6,22	20787	25199
4	6,27	17046	20628
8	6,35	15009	17591
16	6,51	14473	16632
32	6,8	14091	16814
64	7,32	16773	18733
	Imagen agriculture (Landsat)		
Threads	*bps*	T. Cod. (ms)	T. Dec. (ms)
1	3,56	2050	2562
2	3,62	1674	2081
4	3,7	1634	1838
8	3,88	1463	1974
16	4,24	1491	1670
32	4,84	1463	1623
64	5,66	1331	1833

TABLA 5: AUMENTO EN % DE LOS *bps* EN LAS EJECUCIONES CON LUT CON RESPECTO A LA EJECUCIÓN CON 1 THREAD, UTILIZANDO LOS DATOS DE *bps* DE LAS TABLAS 3 Y 4

Threads	Imagen sc03	Imagen agriculture
1	0,00	0,00
2	0,67	1,85
4	1,40	3,97
8	2,81	9,17
16	5,35	19,23
32	9,97	36,04
64	18,47	59,05

TABLA 6: RESULTADOS DE LAS EJECUCIONES DEL SENSOR AVIRIS

Con LUT			
Img.	*bps*	T. Cod. (ms)	T. Dec. (ms)
sc00	6,494	15906	17903
sc03	6,353	15181	16842
sc10	5,837	14890	17034
sc11	6,076	14552	17540
sc18	6,597	15265	17643
Sin LUT			
Threads	*bps*	T. Cod. (ms)	T. Dec. (ms)
sc00	6,192	21638	22761
sc03	6,062	21281	22092
sc10	5,534	20577	21781
sc11	5,810	20745	21516
sc18	6,185	22124	22371
Con LUT vs Sin LUT			
Threads	*bps*	T. Cod. (ms)	T. Dec. (ms)
sc00	4,88 %	-26,49 %	-21,34 %
sc03	4,79 %	-28,66 %	-23,76 %
sc10	5,48 %	-27,64 %	-21,79 %
sc11	4,58 %	-29,85 %	-18,48 %
sc18	6,66 %	-31,00 %	-21,13 %

5.3. Análisis de resultados para imágenes de teledetección

En el primer caso estudiamos el comportamiento de las imágenes del sensor AVIRIS (Tabla 1) con una LUT generada por la imagen *sc03* (elegida al azar). Tras generar la LUT se hicieron las ejecuciones con 8 threads para cada una de las imágenes obteniendo los resultados de la Tabla 6.

Podemos ver que la penalización en *bps* está entre el 4,5 % y el 6,7 %. Si miramos las mejoras de tiempo vemos que en la codificación tenemos mejoras de tiempo que reducen la ejecución entre un 26 % y un 31 %, es cierto que en cuanto a tiempo de decodificación, aunque la mejora es menor, es mucho mayor en proporción a la penalización del aumento de *bps*.

En la Tabla 7 tenemos los resultados de las pruebas realizadas con las imágenes del sensor Landsat. Las ejecuciones con LUT han sido con una LUT generada por la imagen *agriculture*. En este caso vemos que la penalización en cuanto a *bps* es mucho mayor y el tiempo en el que se reduce la ejecución no es suficiente para poder asumir el aumento de *bps*.

Comparando estos datos podemos llegar a una primera conclusión: el software con la implementación de la LUT obtiene mejores resultados con imágenes hiperespectrales (sensor AVIRIS), es decir, con un gran número de bandas. En cambio no obtenemos tales beneficios con las imágenes multiespectrales (sensor Landsat).

5.4. Imágenes médicas

Tras estudiar las imágenes de sensores, que es para lo que está destinado principalmente este software, vamos a estu-

TABLA 7: RESULTADOS DE LAS EJECUCIONES DEL SENSOR LANDSAT

Con LUT			
Img.	*bps*	T. Cod. (ms)	T. Dec. (ms)
agriculture	3,883	1654	1903
coast	3,038	1417	1737
mountain	4,332	1375	1768
Sin LUT			
Threads	*bps*	T. Cod. (ms)	T. Dec. (ms)
agriculture	3,360	1752	2087
coast	2,725	1911	1957
mountain	3,791	1866	1906
Con LUT vs Sin LUT			
Threads	*bps*	T. Cod. (ms)	T. Dec. (ms)
agriculture	15,54 %	-5,59 %	-8,82 %
coast	11,49 %	-25,85 %	-11,24 %
mountain	14,28 %	-26,31 %	-7,24 %

diar el comportamiento del mismo con otro tipo de imágenes. En este caso vamos a utilizar angiografías de corazón y tomografías computacionales. Estas imágenes no son ni hiperespectrales ni multiespectrales, se trata de imágenes monocromáticas, donde cada banda la podemos entender como un frame de un video. Este conjunto de pruebas, al tratarse de imágenes monocromáticas es muy visual, ya que no hay bandas que el ojo no pueda ver, por lo tanto vamos a poder analizar completamente el funcionamiento e impacto de la LUT. En primer lugar tenemos las tomografías computacionales (Tabla 8), son imágenes de 512 por 512 con un número de bandas variable. La LUT ha sido generada con la Imagen 1.

Si observamos la Tabla 9, este conjunto de pruebas nos confirma la conclusión que hemos obtenido previamente. Las imágenes 1 y 2 son imágenes con muchas más bandas que las imágenes 3, 4 y 5. Como podemos ver tenemos una mejora de tiempo algo inferior pero las penalizaciones en bps son mucho menores cuando tenemos un número elevado de bandas.

Ahora miramos las angiografías de corazón (Tabla 10), son imágenes de 512 por 512 con un número de bandas variable. La LUT ha sido generada con la Imagen 1.

En este caso, a simple vista, la Imagen 2 llama la atención, tanto el la banda o "frame" como los resultados de las ejecuciones, en la Tabla 11. Podemos ver que el aumento de bps es desproporcionado. Esto se debe a que es una imagen muy diferente a la imagen con la que se ha generado la LUT, en este caso la Imagen 1. Como estas imágenes son monocromáticas podemos ver a simple vista el impacto que provoca una LUT incorrecta. Esto nos lleva a la conclusión de que es necesario que las LUT se generen y utilicen para imágenes similares, ya que un uso incorrecto puede provocar que el software no sea eficiente.

6 VÍAS DE CONTINUACIÓN

Volviendo a las imágenes hiperespectrales y multiespectrales, gracias los datos que hemos obtenido con las imáge-

TABLA 8: IMÁGENES DE TOMOGRAFÍAS COMPUTACIONALES

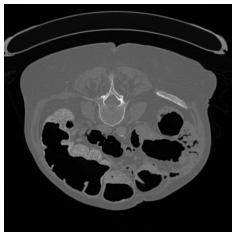

Imagen 1: **596** bandas

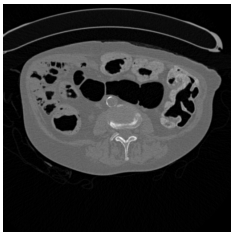

Imagen 2: **637** bandas

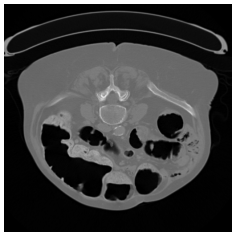

Imagen 3: **82** bandas

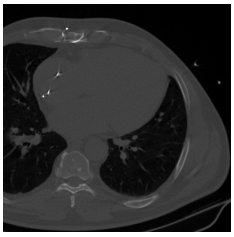

Imagen 4: **54** bandas

Imagen 5: **69** bandas

TABLA 9: RESULTADOS DE LAS EJECUCIONES CON TOMOGRAFÍAS COMPUTACIONALES

Con LUT			
Img.	*bps*	T. Cod. (ms)	T. Dec. (ms)
1	6,154	30039	34721
2	6,045	32773	36334
3	5,804	4227	5032
4	7,019	2913	3648
5	5,056	3485	4243
Sin LUT			
Threads	*bps*	T. Cod. (ms)	T. Dec. (ms)
1	5,943	40491	42835
2	5,875	42005	43293
3	5,514	5994	6290
4	6,746	4375	4351
5	4,760	5589	5632
Con LUT vs Sin LUT			
Threads	*bps*	T. Cod. (ms)	T. Dec. (ms)
1	3,55 %	-25,81 %	-18,94 %
2	2,90 %	-21,98 %	-16,07 %
3	5,27 %	-29,48 %	-20,00 %
4	4,05 %	-33,42 %	-16,16 %
5	6,21 %	-37,65 %	-24,66 %

TABLA 10: IMÁGENES DE ANGIOGRAFÍAS DE CORAZÓN

Imagen 1: **151** bandas

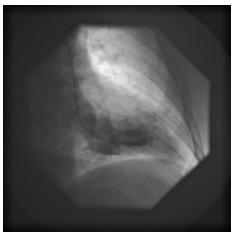

Imagen 2: **271** bandas

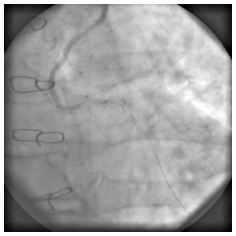

Imagen 3: **186** bandas

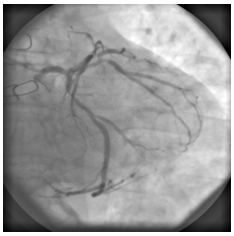

Imagen 4: **97** bandas

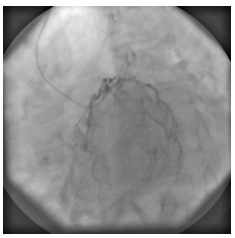

Imagen 5: **83** bandas

TABLA 11: RESULTADOS DE LAS EJECUCIONES CON TOMOGRAFÍAS COMPUTACIONALES

Con LUT			
Img.	*bps*	T. Cod. (ms)	T. Dec. (ms)
1	6,370	7508	9062
2	7,862	13857	16534
3	6,419	9390	11273
4	6,608	5000	6160
5	6,374	4243	5166
Sin LUT			
Threads	*bps*	T. Cod. (ms)	T. Dec. (ms)
1	6,093	9842	10401
2	6,639	18250	18449
3	6,128	12643	12919
4	6,164	6557	7046
5	6,110	5764	5827
Con LUT vs Sin LUT			
Threads	*bps*	T. Cod. (ms)	T. Dec. (ms)
1	4,55 %	-23,71 %	-12,87 %
2	18,41 %	-24,07 %	-10,38 %
3	4,74 %	-25,73 %	-12,74 %
4	7,20 %	-23,75 %	-12,57 %
5	4,33 %	-26,39 %	-11,34 %

nes monocromáticas, sabemos que lo más óptimo es que la LUT que se utilice para codificar las imágenes de los sensores sea una LUT dinámica. Aquí surgen 2 opciones.

1. La primera sería que el sensor cada tiempo T utilice una imagen para generar una nueva LUT que se utilice para codificar las siguientes imágenes durante el tiempo T hasta que se genere la siguiente LUT. Aquí puede surgir el problema de que, por casualidad, se genere una LUT con una imagen defectuosa, lo que provocará compresiones subóptimas durante el siguiente tiempo T. Para solventar esto hay 3 opciones:

   a) Cuando se genere una nueva LUT, mediante un algoritmo se decida si la nueva LUT es correcta o es defectuosa, comparando los valores de la LUT actual y la nueva LUT.

   b) Cada tiempo T generar una nueva LUT pero que esta no sustituya a la anterior, sinó que mediante un algoritmo fusione ambas para obtener una LUT combinada y así corregir los posibles errores al generar LUTs con imágenes defectuosas.

   c) Cada tiempo T se generan N LUTs utilizando las siguientes N imágenes, con ellas se genera una LUT final a partir de la media aritmética de las N LUTs. Esta LUT final sustituye a la anterior LUT.

2. La segunda opción es que los sensores tengan un conjunto de LUTs por coordenadas terrestres. El sensor al tomar una imagen nueva detecta en qué latitud y longitud se encuentra y utiliza la LUT asignada a estas coordenadas. Para este caso habría que dividir el globo en una cuadrícula. Se podría utilizar la cuadrícula actual generada por los meridianos y los paralelos o realizar una cuadrícula con divisiones más pequeñas o más grandes. En una segunda fase se podrían realizar divisiones que no fuesen a partir de una cuadrícula, sinó con formas que se adaptarán más a los accidentes geográficos.

7 CONCLUSIONES

El primer impacto que provoca la implementación de la LUT es la penalización que sufrimos al aumentar el número de threads, como se ha comentado, cuanto más threads estamos asumiendo un error mayor, por lo tanto hay que encontrar el equilibrio entre un aumento de *bps* razonable y una buena mejora de tiempo. Esto no sucede con el código original sin utilizar LUT.

Como hemos podido comprobar la implementación de la LUT aporta beneficios siempre que sea utilizada de forma correcta. Las mejoras las encontramos, principalmente, en imágenes hiperespectrales (sensor AVIRIS) o en imágenes con muchas bandas, como en el caso de algunas imágenes médicas.

También es necesario la utilización de una LUT correcta, ya que una LUT errónea provoca que el software no sea eficiente. Esto lo hemos podido comprobar con las imágenes monocromáticas, ya que con las hiperespectrales e hiperespectrales, al tener una gran mayoría de bandas fuera del rango de visión del ojo humano es más complicado analizar esto.

8 DESARROLLO DEL TRABAJO

Inicialmente el trabajo se estructuró con el siguiente listado de tareas:

- Análisis preliminar del CCSDS 123.0-B-1 y de la modificación usando un codificador aritmético.
- Exportar LUT
  - Analizar el contenido exacto de las LUT que se generan con el código proporcionado.
  - Explorar las diferentes clases que proporciona JAVA para exportar datos o si es necesario utilizar una librería externa. Valorar entre todos los casos la opción que más optimice el rendimiento.
  - Exportar los datos de una forma "literal", es decir, que sea posible analizar las LUT con programas externos utilizando el archivo generado. Explorar la opción exportar 2 archivos, una LUT "literal" y una LUT para ser procesada posteriormente, si de esta forma obtenemos un mejor rendimiento en la codificación.
- Importar LUT en codificador y decodificador: lectura de la LUT desde el archivo en el que se ha exportado y sustitución del código que ejecutaba la predicción por el codigo que utiliza la LUT.
  - Implementar en codificador.
  - Implementar en decodificador.
- Optimizaciones y modificaciones en la LUT.
- Pruebas y comparaciones con distintos conjuntos de imágenes de distintos satélites. Pruebas y comparaciones con imágenes médicas.

El TFG se planificó como se puede ver en la Fig 5. Para realizar este diagrama de Gantt se utilizó el software Gantt Project [9]. La fase de estudio previo se inició en agosto y a principios de septiembre ya se empezó a trabajar en las modificaciones. Para asegurar el cumplimiento de los plazos se destinó mucho tiempo al proceso de *Pruebas y comparaciones con distintos conjuntos de imágenes de distintos satélites...*. Si en el futuro se necesitara alguna modificació seria de este proceso del qual se restaría tiempo. También se dejó el mes de enero entero para el desarrollo de la documentación y posibles imprevistos y/o retrasos.

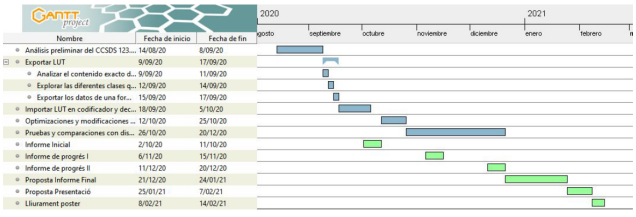

Fig. 5: Diagrama de Gantt inicial

Al acabar el proceso de *Optimizaciones y modificaciones en la LUT* se decidió dar un paso más y optimizar el proceso de codificación. Para ello, se restó tiempo del proceso de *Pruebas y comparaciones con distintos conjuntos de imágenes de distintos satélites*, como se puede ver en la Fig.

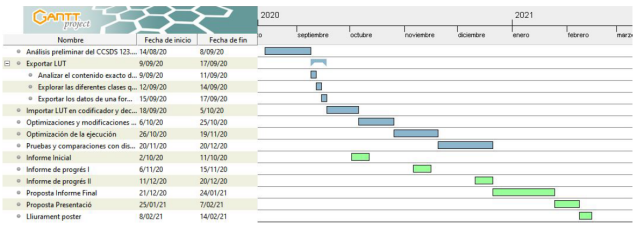

Fig. 6: Diagrama de Gantt modificado con la optimización de la codificación.

6, dejando igualmente el mes de enero para el desarrollo de la documentación y más posibles imprevistos y/o retrasos.

Se han cumplido todos los plazos establecidos en los diagramas de Gantt sin ningún tipo de retraso.

AGRADECIMIENTOS

Agradecer a Joan Bartrina Rapesta, tutor de este Trabajo de Fin de Grado, por sus explicaciones, ayudas y consejos a lo largo de los meses en los que se ha desarrollado este proyecto. Agradecer a mi familia por todo el apoyo durante mi etapa universitaria.

REFERENCIAS

[1] Wikipedia, "Satélite Artificial". Accessed Dec. 30, 2020. [Online]. Available: https://es.wikipedia.org/wiki/Satélite_artificial

[2] Consultative Committee for Space Data Systems (CCSDS). [Online]. Available: http://www.ccsds.org/

[3] Lossless Multispectral and Hyperspectral Image Compression, document CCSDS 123.0-B-1, Blue Book, May 2012.

[4] J. Bartrina-Rapesta, I. Blanes, F. Aulí-Llinàs, J. Serra-Sagristà, V. Sánchez and M.W. Marcellin, "A Lightweight Contextual Arithmetic Coder for On-Board Remote Sensing Data Compression".

[5] GISGeography, "Multispectral vs Hyperspectral Imagery Explained", Aug. 2020. Accessed: Sep. 29, 2020. [Online]. Available: https://gisgeography.com/multispectral-vs-hyperspectral-imagery-explained

[6] F. Aulí-Llinàs, "Context-adaptive Binary Arithmetic Coding with Fixed-length Codewords".

[7] NASA, AVIRIS - Airborne Visible / Infrared Imaging Spectrometer. , Accessed: Dec. 14, 2020. [Online]. Available: https://aviris.jpl.nasa.gov/aviris/index.html

[8] NASA, Technical Details — Landsat Science, Accessed: Dec. 17, 2020. [Online]. Available: https://landsat.gsfc.nasa.gov/about/technical-details

[9] Gantt Project , Accessed: Sept. 17, 2020. [Online]. Available: https://www.ganttproject.biz/