Point Cloud Compression

Xavier López Lara

Resumen– Este documento aborda temas relevantes en el procesamiento de Point Data Cloud, en concreto con el conjunto de datos KITTI velodyne LiDAR. Se describe el uso de esta tecnología para la captura de datos y su capacidad para obtener información detallada del entorno en 3D. Se aborda el problema del preprocesamiento de datos en point cloud para lograr mejorar las tasas de compresión, poder realizar una representación óptima de la información y el análisis de las nubes de puntos, también se realizan métricas de compresión sobre diferentes conjuntos de datos con distintos algoritmos de compresión como LASzip, FRL y GZIP para analizar los datos obtenidos y sacar conclusiones sobre cuál es más óptimo para la compresión de grandes conjuntos de datos con la finalidad de permitir una mayor eficiencia en el almacenamiento y procesamiento de la información. Sobre FRL se han reflejado diferentes tipos de problemáticas con el dataset KITTI, se busca la alternativa de comprimir KITTI con LASzip, ya que ofrece una compresión más óptima en puntos capturados con la tecnología LiDAR y se comparan los resultados con los datos obtenidos con FRL y GZIP. LASzip es el algoritmo de compresión que se ha analizado en este documento más óptimo para el conjunto de datos KITTI y basados en LiDAR.

Palabras clave– Point Cloud Compression, LiDAR, KITTI, Velodyne, LASzip, GZIP, Fast Run-Length (FRL), Bitrate

Abstract– This document addresses relevant topics in Point Cloud Data Processing, specifically focusing on the KITTI Velodyne LiDAR dataset. It describes the use of this technology for data capture and its ability to obtain detailed 3D information about the environment. The problem of data preprocessing in point cloud is discussed to improve compression rates, achieve optimal representation of information, and analyze point clouds. Compression metrics are applied to different datasets using various compression algorithms such as LASzip, FRL, and GZIP to analyze the obtained data and draw conclusions about the most optimal method for compressing large datasets, aiming to enhance storage and processing efficiency. Different issues with the KITTI dataset using FRL are highlighted, and an alternative of compressing KITTI with LASzip is explored, as it offers superior compression for LiDAR-captured points. The results are compared with those obtained using FRL and GZIP. LASzip is determined to be the most optimal compression algorithm for the KITTI dataset based on LiDAR in this document.

Keywords– Point Cloud Compression, LiDAR, KITTI, Velodyne, LASzip, GZIP, Fast Run-Length (FRL), Bitrate

1 Introducción

Actualmente se hace más necesaria la captura de datos precisos en tiempo real para multitud de aplicaciones y la tecnología Point Cloud ha demostrado ser una herramienta muy útil para el uso de estas aplicaciones. Esta permite capturar y procesar datos tridimensionales muy precisos en tiempo real, el formato de estos puntos están representados por coordenadas XYZ que describen la posición exacta de cada punto en un espacio tridimensional, a este tipo de datos se les denomina; datos geométricos. Los datos geométricos van acompañados de información adicional; los atributos. Estos atributos permiten representar el color, la intensidad, la altura o su reflectividad [1], de cada uno de los puntos representados. Por lo tanto, dependiendo del objetivo, los esquemas de compresión de Point Data Clouds se agrupan en algoritmos de geometría o atributos. Los Point Data Cloud se crean mediante el empleo de sistemas de adquisición de datos 3D, como escáneres láser o cámaras 3D, que registran información en alta resolución sobre la posición y la forma de un objeto o una escena. Luego, estos puntos se almacenan en un archivo en formato de Point Data Cloud.

• E-mail de contacto: 1498102@uab.cat
• Mención realizada: Tecnologías de la Información
• Trabajo tutorizado por:Joan Serra Sagristà (departament)
• Curs 2022/2023

En muchos casos, debido a la gran cantidad de información, los point clouds son muy pesados y realizar una compresión adecuada que se adapte al tipo de Point Cloud puede suponer una mejora para poder manejar la información de una manera más liviana. Para ello en el siguiente documento se abordan diferentes técnicas de compresión sobre distintos tipos de datasets con características específicas, dichos datasets en ocasiones tendran que ser tratados previamente para una mayor eficiencia en la compresión. Todos los experimentos efectuados están principalmente basados sobre los datos geométricos.

Un paso estándar de preprocesamiento, después de adquirir los datos de la nube de puntos, es la cuantificación de su información de geometría en coordenadas enteras [2] . Este proceso se conoce como voxelización, donde cada punto en el espacio 3D se representa como una celda cúbica llamada voxel y se coloca en una rejilla 3D. Se dice que un voxel está ocupado cuando corresponde a un punto de la nube de puntos, en contraste con los voxels no ocupados que forman el espacio vacío. Aunque la voxelización de una nube de puntos reduce en cierta medida su información de geometría original, se requiere un procesamiento adicional para una compresión significativa.

Principalmente, al ejecutar una compresión de información y dependiendo del objetivo se puede realizar con pérdidas o sin pérdidas. La compresión con pérdida de Point Data Clouds puede lograr tasas de bits relativamente bajas a expensas de introducir distorsión en los datos descomprimidos, mientras que los esquemas de compresión sin pérdida preservan perfectamente los datos originales a costa de una tasa más alta. Todos los resultados que se verán a lo largo del documento están efectuados con algoritmos de compresión sin pérdidas, por lo que mantendremos la integridad de los datos en el proceso de compresión como de descompresión del mismo.

2 OBJETIVOS

Durante toda la fase de investigación se trabaja con distintos tipos de conjuntos de datos, más adelante se explican los datasets con los que se ha trabajado. Sin embargo, este artículo se centrará en profundizar sobre el dataset KITTI Velodyne [3] y se analizará detalladamente explorando sus características y peculiaridades.

Se llevarán a cabo experimentos utilizando el algoritmo de compresión Fast Run-Length (FRL) . Estas pruebas se realizarán con datasets de menor tamaño para no demorar tiempo, ya que se realizaran con el fin de comprender el funcionamiento de FRL y analizar cómo se podría comportar con el dataset KITTI Velodyne y permitir obtener una comprensión sobre el preprocesamiento del dataset y el posible rendimiento del algoritmo en relación con este conjunto de datos específico.

A continuación se deberá efectuar un preprocesamiento de los datos del dataset KITTI Velodyne. El objetivo de este preprocesamiento es preparar los datos para poder hacer compresiones en muestras aleatorias de ejemplo.

La finalidad de los experimentos y el trabajo efectuado es reducir el tamaño de los datasets proporcionados mediante la aplicación de algoritmos de compresión sin pérdidas. Además, se realizarán ejecuciones con otros algoritmos de compresión como: GZIP [5], LASzip [8] y FRL.

Finalmente, comparar y evaluar el rendimiento de diferentes algoritmos de compresión en los datasets. El analisis de las métricas de compresión y los resultados obtenidos harán posible determinar cuál de los algoritmos de compresión sin pérdidas ofrece los mejores resultados en términos de reducción de tamaño de los datasets.

3 DATASETS

Se van a citar los diferentes conjuntos de datos utilizados para las pruebas, se mostrarán resultados de compresión de los diferentes algoritmos explicando sus resultados y modificaciones realizadas para más tarde poder realizar una comparativa global.

3.1 KITTI Velodyne

El conjunto de datos KITTI son capturados gracias a una plataforma de conducción autónoma como se muestra en la figura 1. Las tareas de dicho coche autonomo son capturar: flujo óptico, odometría visual, detección de objetos 3D y seguimiento 3D. Para ello, se equipa un automóvil familiar estándar con dos cámaras de video de alta resolución en color y en escala de grises. La precisión de la verdad de referencia se proporciona mediante un escáner láser Velodyne y un sistema de localización GPS. Los conjuntos de datos se capturan conduciendo alrededor de la ciudad mediana de Karlsruhe, en áreas rurales y en autopistas. Se pueden ver hasta 15 automóviles y 30 peatones por imagen. El conjunto de datos se compone de 7518 point clouds capturados con la tecnología Light Detection and Ranging (LiDAR) que es una técnica similar al radar que utiliza láser en lugar de ondas de radio y proporciona una nube de puntos con mediciones de distancia, un entorno. El funcionamiento de un LiDAR comienza con la emisión de un pulso láser, seguido por la captura del láser reflejado mediante la medición del tiempo que tarda el láser. Los datos han sido almacenados en una matriz de Nx4 en formato binario, dichos datos se han almacenado alineados por filas, por lo que las mediciones contienen la siguiente información:

- Los 3 primeros valores indican los datos geométricos: X,Y,Z.
- El último valor contiene la información de reflectancia de los puntos.

3.1.1 Procesamiento y visualización

Para procesar los datos es necesario hacer la transformación del fichero de formato binario a ascii. Para ello se ha desarrollado un conversor de binario a ascii programado en Python, este recibe un archivo en formato binario y lo transforma a ascii. Como se ha mencionado anteriormente, los datos contienen N*4 valores, por lo que los primeros 4 valores se refieren a la primera medición.

Se convierte el dataset por completo y se hace un procesamiento de los datos del dataset para poder comprimirlo con FRL. Para ello se transforman los valores negativos en positivos, se eliminan los decimales, se ordenan los valores, se eliminan los repetidos, se indica a FRL el tamaño de los voxels..

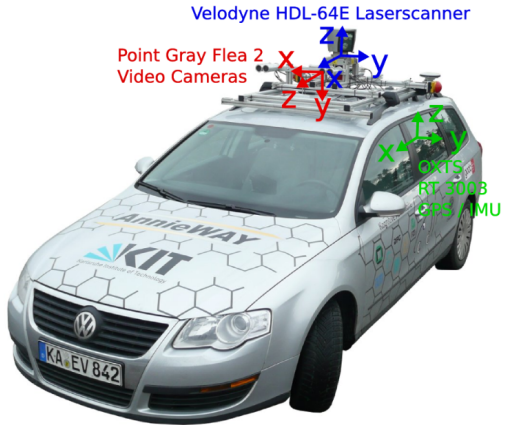

Fig. 1: Esta figura muestra el vehículo completamente equipado.

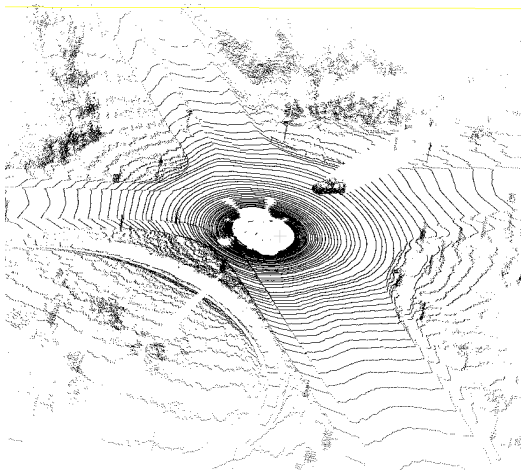

Fig. 2: Visualización de un Point Cloud de KITTI

Como se puede observar en la figura 2 se ve una representación gráfica de un point cloud, todos los point clouds del dataset son similares, en ellos se puede observar un conjunto de puntos más densos en la parte central del central de la imagen que se van esparciendo de manera más gradual a medida que se aleja del punto central en el que se encuentra el sensor láser.

TABLA 1: ESTADÍSTICAS - KITTI VELODYNE

Point Cloud	Num. Points	Range x	Range y	Range z
000217.ply	127827	158469	129937	30562
000571.ply	127802	158896	152269	20077
000775.ply	127836	156458	92296	24518
001500.ply	127891	158282	153626	24306
002414.ply	127936	157889	157355	26937
002648.ply	127977	159715	155019	22676
003959.ply	127701	143757	76002	7838
004331.ply	127976	157548	125332	19592
005091.ply	127978	158899	147022	11128
006587.ply	128010	158247	73518	30542

La entropía brinda información importante a la hora de afrontar la compresión del dataset. El eje de la Z tiene menor valor en la entropía en todos los casos, en ocasiones es muy cercana a 0 como podemos observar en la tabla 2 esto nos indica que la entropía es baja y como se puede ver en las figuras 3 y 4 vemos que los puntos en el eje de la Z son más densos que en X o Y. Es relevante destacar que la entropía de la tabla 2 hace referencia a la entropía del *signaling rectangle* [7]. Un *signaling rectangle* es desde donde comienza a haber información hasta que se en cuenta el último punto con datos, al aplicar esta técnica se consigue un ”rectánguloçon los puntos que contienen datos y los que no contienen datos se vuelven redundantes, ya que no tienen información. Cuanto mayor sea este rectángulo proporcionará información que nos indicará que los puntos están más dispersos sobre las coordenadas por la parte contraria, cuanto menor sea dicho triángulo, los puntos quedarán más concentrados. Por lo tanto, si se aplica dicha técnica y el resultado es un rectángulo “pequeño” solo se debe comprimir dicha información y, por lo tanto, se optimiza el proceso de compresión.

TABLA 2: ENTROPÍA DE ”SIGNALING RECTANGLE- KITTI VELODYNE

Point cloud	SR entropia x	SR Entropia y	SR Entropia z
000217.ply	0.5665	0.4689	0.1330
000571.ply	0.6500	0.2761	0.2253
000775.ply	0.5665	0.5435	0.1274
001500.ply	0.4138	0.5916	0.2222
002414.ply	0.6500	0.6840	0.0597
002648.ply	0.4854	0.6840	0.0922
003959.ply	0.5435	0.9182	0.1623
004331.ply	0.6500	0.7642	0.0950
005091.ply	0.6501	0.6193	0.0974
006587.ply	0.5435	0.5032	0.0384

Se puede ver el que el número máximo de voxels es de aproximadamente 128000 para todas las muestras, dichas muestras no son secuenciales, es decir, se han escogido las muestras de manera que como se puede observar hay varianza entre cada una de ellas, por lo que los resultados deberían variar, pero se puede observar que el dataset es bastante similar en todos los casos. Teniendo esto en cuenta; el rango máximo en el eje de las X es de, 159715, 157355 para la Y, 30562 para la Z. Estos datos nos ofrecen un contexto sobre el dataset muy importante; el rango de la coordenada Z es un 80% aproximadamente menor que el de las coordenadas XY por lo que a la hora de realizar la compresión es importante tener en cuenta el este valor, ya que podemos aplicar técnicas de procesamiento de los datos sobre las coordenadas para intentar mejorar la compresión. El eje XY es mucho más esparcido que el de Z, esto es lógico porque el eje Z nos está marcando la altura y en el ámbito en el que se recolectan dichos datos no está centrado en dicho eje porque en la conducción autónoma se centra en los datos que se obtienen en las coordenadas XY. Las figuras 3, 4 y 5 muestran de manera gráfica el tipo de point cloud desde una perspectiva 2D. Observando de manera rápida los puntos pueden quedar más dispersos o más densos según la posición en la que ordenemos los puntos, por lo que es un factor a tener en cuenta a la hora de utilizar diferentes técnicas para comprimir.

TABLA 3: OCUPACIÓN DE VOXELS - KITTI VELODYNE

Point Cloud	Ocup. voxels x	Ocup. voxels y	Ocup. voxels z
000217.ply	23	84	320
000571.ply	19	32	356
000775.ply	30	249	292
001500.ply	19	21	392
002414.ply	17	25	444
002648.ply	17	37	425
003959.ply	28	71	386
004331.ply	21	96	434
005091.ply	23	91	186
006587.ply	16	54	523

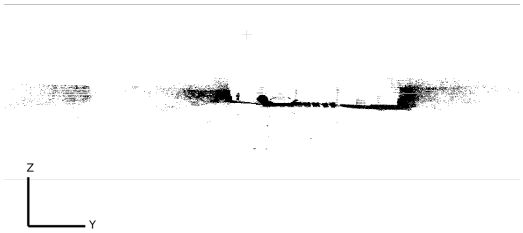

Fig. 3: Representación Point Cloud KITTI en 2D sobre el eje de la X

Fig. 4: Representación Point Cloud KITTI en 2D sobre el eje de la Y

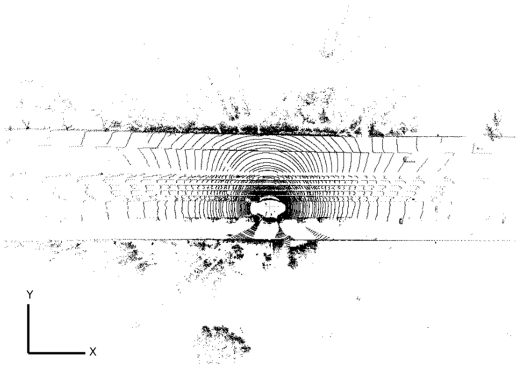

Fig. 5: Representación Point Cloud KITTI en 2D sobre el eje de la Z

Fig. 6: Muestra del *point cloud basketball* MPEG

Fig. 7: Muestra del *point cloud dancer* MPEG

3.2 MPEG Dataset

Este un conjunto de datos [8] que se utiliza en el campo de la visión por computador para el estudio y análisis de secuencias de mallas tridimensionales de seres humanos en movimiento. Este conjunto de datos se caracteriza por contener modelos de mallas de alta resolución que representan la forma y textura de diferentes personas realizando una variedad de movimientos y acciones, además puede contener anotaciones adicionales, como la posición de las articulaciones y la correspondencia entre los modelos de malla y los puntos de referencia anatómicos.

4 ALGORITMOS DE COMPRESIÓN

Seguidamente, se explican los diferentes algoritmos de compresión con los que se ha trabajado en la fase experimental para obtener los resultados.

4.1 Algoritmo FRL

El algoritmo FRL opera directamente en el dominio de voxels, por lo que va codificando secuencias de voxels a través de un codificador aritmético, según el contexto de dicho voxel por lo que se codifican secuencias de voxels, para conseguir esto se requiere de un preprocesamiento previo que ordene el point cloud en el eje que se desee. El algoritmo se compone de dos fases:

En la primera fase actuan se reciben como entradas las coordenadas de los voxels ocupados de la nube de puntos. Se escanea la información de geometría de la nube de puntos, formando ejecuciones de voxels que comienzan y terminan dentro de un único plano 2D del espacio de voxels llamado ”slice”[9]. Cada ejecución está compuesta por todos los voxels no ocupados que están posicionados después del final de la ejecución anterior y un voxel ocupado. Es decir, la ejecución termina tan pronto como se encuentra el primer voxel ocupado siguiendo el orden de exploración por rasterización. El proceso concluye devolviendo las coordenadas de los voxels que representan el inicio y el final de cada ejecución. A continuación se le asigna el estado de cada voxel en un diccionario que su tamaño está ordenado según los voxels ocupados y cada uno de sus elementos se accede solo una vez durante la compresión del point cloud. En la segunda fase se ejecutan los procesos de cálculo y codificación aritmética.

4.2 GZIP

GZIP es un algoritmo de compresión sin pérdidas que está basado en *DEFLATE* [10], en el cual se combina el uso de LZ77 [11] y Codificación Huffman[12], funciona dividiendo los datos de entrada en bloques de 8 bytes. Cada bloque se comprime utilizando una combinación de dos técnicas:

- **LZ77**: Esta técnica busca secuencias repetidas de bytes en el bloque y las almacena como una referencia a la aparición previa de la secuencia. Esto puede reducir significativamente el tamaño del bloque, especialmente si contiene muchos patrones de repetición de valores.
- **Codificación Huffman**: Esta técnica asigna códigos de longitud variable a cada byte en el bloque, según la frecuencia con la que aparece cada byte. Esto puede reducir aún más el tamaño del bloque aprovechando el hecho de que algunos bytes son más comunes que otros.

GZIP admite 10 niveles de compresión diferentes, del 1 al 9. Cuanto mayor sea el nivel de compresión, más tiempo tomará el algoritmo para comprimir los datos, pero el archivo comprimido será más pequeño. El nivel de compresión 6 es el predeterminado dado que es el más equilibrado entre la relación de compresión y velocidad.

Dependiendo del nivel de compresión elegido GZIP actua con diferentes métodos para lograr realizar la compresión de los datos, a continuación se describe el funcionamiento de los niveles de compresión 1, 6 y 9 que son los que se han utilizado durante el proyecto:

- **Nivel de compresión 1**: El nivel de compresión más rápido. El algoritmo solo utiliza compresión LZ77 y no utiliza la codificación Huffman.
- **Nivel de compresión 6**: El nivel 6 usa una combinación de compresión LZ77 y codificación Huffman. El algoritmo usa compresión LZ77 para encontrar secuencias repetidas de bytes, y luego utiliza codificación Huffman para asignar códigos de longitud variable a cada byte. A diferencia del nivel 1, este ofrece un resultado en la compresión mayor, por lo que el resultado del fichero sería más bajo, por contra el tiempo dedicado para realizar la compresión es mayor en dicho nivel.
- **Nivel de compresión 9**: Utiliza la compresión LZ77 de forma más óptima y la codificación Huffman más compleja. El algoritmo LZ77 empleado en este nivel es más agresivo. Esto significa que encontrará más secuencias repetidas de bytes, Por lo que el tamaño de archivo comprimido será menor, pero también llevará más tiempo comprimir los datos.

4.3 LASzip

LASzip se enfoca especialmente en la compresión para los tipos de puntos LiDAR [13]. En lugar de comprimir directamente los datos geométricos, utiliza técnicas de predicción y codificación de entropía para comprimir las diferencias entre las coordenadas de puntos consecutivos.

Para las coordenadas XY, predice las diferencias de dichas coordenadas entre el punto anterior y el punto actual, basándose en las diferencias de coordenadas de los cinco puntos inmediatamente anteriores con el mismo tipo de resultado. La predicción se realiza empleando la mediana de estas diferencias. Al usar la mediana, se tiene en cuenta la distribución de las diferencias de coordenadas y se obtiene una mejor estimación de la diferencia actual.

Para la coordenada Z, LASzip usa predicción sobre el valor del punto actual basándose en el valor del punto inmediatamente anterior con el mismo resultado. La predicción se efectúa tomando el valor del punto anterior como el valor predicho.

Una vez que se han llevado a cabo las predicciones de las diferencias de coordenadas, LASzip emplea técnicas de codificación de entropía para codificar estas diferencias y reducir aún más el tamaño de los datos. La codificación de entropía se basa en la frecuencia de ocurrencia de resultados de las diferencias. Los valores más comunes se codifican con menos bits, mientras que los valores menos comunes se codifican con más bits.

Además de la compresión de coordenadas, LASzip también utiliza compresión para otros atributos [14] de los puntos de datos, como los valores RGB (rojo, verde, azul) y el tiempo GPS. Emplea técnicas similares de predicción y codificación de entropía para comprimir estos atributos, pero no se va a entrar en detalle sobre la compresión de los atributos.

5 ESTADO DEL ARTE

La compresión de nubes de puntos es un área de investigación activa en el campo de la visualización y procesamiento de datos tridimensionales. La creciente disponibilidad de dispositivos de escaneo 3D y la cantidad de aplicaciones que generan grandes cantidades de datos de nubes de puntos ha llevado a la necesidad de desarrollar técnicas eficientes de compresión enfocadas a los distintos patrones que siguen las nubes de puntos. El objetivo principal de la compresión de las mismas es reducir el tamaño de los datos sin perder información crucial para su posterior análisis y visualización.

En los últimos años, se han propuesto varias técnicas de compresión específicas para nubes de puntos, cada una con sus ventajas y desafíos particulares. Entre las técnicas más comunes se encuentran las basadas en geometría, atributos y esquemas de predicción. Las técnicas basadas en geometría se centran en reducir la redundancia espacial, aprovechando la información de la estructura y la conectividad de los puntos. Estas técnicas incluyen métodos como la simplificación de mallas, los algoritmos de octree y las representaciones basadas en voxels entre otras.

- Compresión basada en octree [15]: Este método divide la nube de puntos en una estructura jerárquica de octree y luego codifica cada nodo en el octree utilizando un código binario. La compresión basada en octree es relativamente eficiente y puede lograr buenas tasas de compresión, pero puede ser sensible al ruido y a los valores atípicos en la nube de puntos.
- Compresión basada en wavelet[16]: Este método descompone la nube de puntos en un conjunto de coefici-

entes wavelet y luego codifica cada coeficiente empleando un esquema de cuantización y codificación de entropía. La compresión basada en wavelet puede lograr altas tasas de compresión y es relativamente insensible al ruido, pero puede ser computacionalmente costosa.

- Compresión basada en aprendizaje profundo [17]: Este método usa redes neuronales profundas para aprender una representación comprimida de la nube de puntos. La compresión basada en aprendizaje profundo puede lograr tasas de compresión de vanguardia, pero puede ser computacionalmente costosa para entrenar la red neuronal.

Por otro lado, las técnicas basadas en atributos se centran en la reducción de la redundancia en los datos de atributos asociados a los puntos, como el color, la intensidad o la reflectancia. Estas técnicas incluyen métodos de codificación predictiva, cuantización adaptadas a las características específicas de los atributos.

Además de las técnicas mencionadas, también han surgido enfoques híbridos que combinan tanto la información geométrica como los atributos para lograr una mayor eficiencia de compresión. Estos enfoques suelen utilizar esquemas de codificación adaptativa y métodos de segmentación para identificar y codificar regiones con características similares.

Es importante mencionar que la evaluación de las técnicas de compresión de nubes de puntos se realiza mediante métricas como la tasa de compresión, la calidad de reconstrucción, el tiempo de codificación/descodificación y la eficiencia de almacenamiento. Además, se considera la escalabilidad de los algoritmos para lidiar con grandes conjuntos de datos y su capacidad para mantener la fidelidad de la información relevante durante la compresión.

En el campo de la compresión de nubes de puntos, existen áreas en las que hay una falta de investigación o conocimiento insuficiente. Algunas de estas áreas incluyen la compresión adaptativa, que busca ajustar la tasa de compresión según la complejidad de la nube de puntos; la preservación de características semánticas, como el color y las texturas de los puntos; la compresión de nubes de puntos dinámicas, que involucran cambios en los puntos a lo largo del tiempo [18]; la evaluación objetiva de la calidad de la compresión, donde aún no hay consenso sobre cuáles son las mejores métricas; y la compresión de nubes de puntos masivas, esta plantea desafíos en términos de eficiencia computacional y almacenamiento. Hay muchas áreas que ofrecen oportunidades de investigación importantes para mejorar las técnicas de compresión y las aplicaciones.

6 RESULTADOS

Se realizan las pruebas de compresión con los algoritmos de compresión sobre los datasets descritos anteriormente. En la primera tabla se muestran los resultados tras comprimir el dataset con las con los datos geométricos en orden XYZ [20] de las muestras del dataset KITTI.

Como se puede observar en la tabla 4 los resultados son ampliamente superiores en cuanto al ratio de compresión sobre LASzip respecto a cualquier nivel de compresión utilizado en LASzip. LASzip supone una mejora de aproximadamente un x2.13 respecto a GZIP nivel 9. Hay que tener en cuenta que el dataset KITTI usa la tecnología LiDAR por lo que era predecible que LASzip obtendría mejores resultados en este tipo de dataset. Esto se debe a que LASzip está diseñado exclusivamente para tratar con nubes de puntos capturados con la técnología LiDAR, aunque es probable que esto pueda variar dependiendo de la cantidad de puntos y lo distribuidos que estén en el dataset. FRL ha sido ejecutado con KITTI, pero los datos obtenidos no se han introducido en la tabla 4 dado que no se ha podido verificar la integridad de los resultados.

TABLA 4: RATIO DE COMPRESIÓN - GZIP Y LASZIP

Point cloud	tamaño del archivo comprimido [KB]				ratio compresión			
	LAZ	GZIP-1	GZIP-6	GZIP-9	LAZ	GZIP-1	GZIP-6	GZIP-9
217	452,949	1006,564	890,872	886,714	5,64	2,41	2,73	2,74
571	533,116	1032,121	933,39	931,183	4,79	2,36	2,61	2,62
775	347,247	974,645	862,506	860,88	7,36	2,49	2,82	2,82
1500	452,265	1016,222	916,366	914,602	5,66	2,4	2,66	2,66
2414	405,152	1003,139	899,497	897,248	6,32	2,43	2,71	2,72
2648	426,266	1011,122	905,329	901,699	6,01	2,41	2,7	2,71
3959	568,075	960,494	856,989	852,83	4,50	2,39	2,68	2,69
4331	469,456	1018,444	910,98	908,108	5,45	2,4	2,68	2,69
5091	451,246	985,57	888,168	885,713	5,67	2,34	2,6	2,61
6587	419,957	990,884	878,883	875,679	6,10	2,46	2,77	2,78
total	**4525,729**	**9999,205**	**8942,98**	**8914,656**	**5,75**	**2,41**	**2,70**	**2,70**

TABLA 5: TIEMPO DE COMPRESIÓN Y DESCOMPRESIÓN - GZIP, FRL Y LASZIP

Point cloud	tiempo compresión [seg]					tiempo descompresión [seg]			
	LAZ	FRL	GZIP-1	GZIP-6	GZIP-9	LAZ	GZIP-1	GZIP-6	GZIP-9
217	1,0248	12,138	0,0877	0,2986	0,5581	1,0191	0,0257	0,0236	0,0278
571	1,0391	10,6883	0,0671	0,2937	0,5369	1,0148	0,0249	0,0255	0,0233
775	1,0261	14,4112	0,0701	0,2868	0,5111	1,0344	0,0239	0,0230	0,0231
1500	1,0141	12,0177	0,0694	0,2968	0,5656	1,0127	0,0287	0,0261	0,0229
2414	1,0246	14,5308	0,0689	0,2863	0,5962	1,0180	0,0252	0,0229	0,0293
2648	1,0206	11,9621	0,0801	0,2990	0,5535	1,0223	0,0253	0,0232	0,0231
3959	1,0198	3,13652	0,0776	0,2881	0,4266	1,0268	0,0240	0,0269	0,0269
4331	1,0170	8,40686	0,0774	0,3230	0,5229	1,0206	0,0269	0,0230	0,0229
5091	1,0336	6,36127	0,0780	0,3125	0,4084	1,0238	0,0241	0,0211	0,0231
6587	1,0204	5,26054	0,0858	0,3097	0,5684	1,0204	0,0248	0,0257	0,0227
total	**1,0240**	**9,8913**	**0,0762**	**0,2995**	**0,5248**	**1,0213**	**0,0253**	**0,0241**	**0,0245**

La tabla 5 hace referencia al tiempo de compresión por cada una de las muestras. Priorizando el tiempo GZIP en su versión más rápida es hasta x13.4 veces más rápido que LASzip y x130 veces más rápido que FRL. En este caso el tiempo de compresión es relevante porque el conjunto de datos KITTI tiene un tamaño aproximado de 30 GB, con LASzip se tarda aproximadamente unos 15 minutos, 2 horas con FRL y aproximadamente 1 minuto con GZIP-1.

La tasa de bits por ocupación de voxel representa la cantidad promedio de bits necesarios para representar cada punto en la nube de puntos después de aplicar un algoritmo de compresión. Este valor se desea que sea mínimo, ya que eso indica que se ha conseguido una mayor compresión.

En la tabla 7 se han comprimido los datasets MPEG y OpenTopography [21]. La cantidad de puntos de estos datasets es muy superior a los de KITTI, Es importante observar que OpenTopography cumple claramente los requerimientos que LASzip necesita para sacar todo su potencia de compresión y es que dispone de casi 30 millones de puntos, lo que indica que muchos de estos puntos se van a repetir en sus primeros digitos, entonces dichos numeros se guardarán en la cabezera del archivo y los puntos serán de menor tamaño, esto ayuda a conseguir una menor compresión y a su vez trabajar con datos más pequeños por lo que la tarea de comprimir tantos puntos será mucho más liviana. Todo esto se refleja en que LASzip obtiene un ”bitrate”y un tiempo de

compresión claramente superior al de los otros algoritmos. En este caso, FRL no es capaz de realizar una compresión de OpenTopography. Esto es debido al tamaño de los datos, al hacer los cálculos de probabilidades y operar con puntos de valor tan grande, estas operaciones se vuelven cada vez más costosas por la parte de memoria. Finalmente, se obtiene un desbordamiento de memoria, realmente este puede llegar a comprimir dicho dataset, pero requiere de modificaciones en el código para poder optimizar el uso de la memoria.

TABLA 6: BITS POR OCUPACIÓN DE VOXEL

Point cloud	Bits por ocupacion de voxel				
	LAZ	FRL	GZIP-1	GZIP-6	GZIP-9
217	28,34	27,13	62,99	55,75	55,49
571	33,37	35,68	64,60	58,42	58,29
775	21,73	24,41	60,99	53,97	53,87
1500	28,29	28,01	63,56	57,32	57,21
2414	25,33	27,53	62,72	56,24	56,10
2648	26,64	29,96	63,20	56,59	56,36
3959	35,58	36,13	60,17	53,68	53,42
4331	29,34	28,40	63,66	56,94	56,76
5091	28,20	27,36	61,60	55,52	55,36
6587	26,24	28,26	61,92	54,92	54,72
total	**28,3109**	**29,2913**	**62,5469**	**55,9400**	**55,7628**

TABLA 7: BITRATE Y TIEMPO DE COMPRESIÓN DATASETS MPEG Y OPENTOPOGRAPHY - GZIP, FRL Y LASZIP

Point cloud	número de puntos	Bitrate [MB/s]			Tiempo compresión [seg]		
		LAZ	FRL	GZIP-6	LAZ	FRL	GZIP-6
shiva	1009132	31,4884135	15,6627	7,79	1,0255544	0,918296	6,59092198
basketball	2925514	42,6871285	1,04817	11,03	2,0427754	1,34036	7,08094683
dancer	2592758	37,9783604	1,16677	9,93	2,0348939	1,0219	6,96568904
mask	272684	8,55580682	19,2417	5,51	1,0199792	0,23511	2,47363137
OpenTopography	29182519	64,3759287	-	9	9,0662864	-	89,480443

7 CONCLUSIONES Y MEJORAS

A lo largo del proyecto se presentaron dificultades con el algoritmo de compresión FRL que se intentaron solucionar, en primera instancia el algoritmo de compresión estaba programado en C y específicamente puesto a prueba con los datasets MPEG. Se migró el código a C++ por lo que así se puede operar con tipos de datos más grandes y evitar los desbordamientos de memoria que sufría operar con los datos. Las ejecuciones mostradas en los resultados son ejecutadas en la última versión de FRL y en este último caso sí que es posible obtener una çompresión"de los datos, pero nada fiable. En el apartado de descompresión de los ficheros, FRL no es capaz de descomprimir el dataset KITTI en su última versión, por lo tanto, no se puede lograr identificar la integridad de los resultados obtenidos.

Durante todo el proyecto se ha enfocado a la compresión sin pérdidas, por dicha razón se ha preferido buscar alternativas de compresión sin perdidas y poder realizar métricas de comparación aplicando la compresión y descompresión sobre los datos. FRL obtiene los mejores resultados con los samples MPEG utilizados, pero, en cambio, tiene dificultades a la hora de comprimir otro tipo de datasets, generalmente en las pruebas ejecutadas LASzip es superior si se realiza un ratio entre tiempo/compresión. Especialmente si hablamos de datos de tipo LiDAR de gran densidad de puntos, ya que en las pruebas realizadas se han realizado compresiones con otro tipo de datasets de tipo LiDAR, pero con una densidad de puntos mucho menor y los resultados obtenidos no son tan óptimos como cabría esperar.

GZIP es el algoritmo más usado a lo largo de la historia y los resultados obtenidos apoyan el porqué de esto; este comprime excepcionalmente rápido y a su vez es capaz de hacerlo con cualquier tipo de fichero sin importar la extensión, también ofrece en su nivel 6 un equilibrio entre compresión y tiempo de compresión. Lamentablemente, no es un algoritmo a tener en cuanto en el ámbito de las nubes de puntos, ya que requiere de utilizar técnicas enfocadas a los patrones que puede seguir un point cloud.

La transformación de los datos de binario a ascii del dataset KITTI fue un poco confuso para comprobar la integridad de los datos transformados, se tuvieron que efectuar varias comprobaciones sobre el pesado dataset lo que ralentizó el trabajo.

Con más tiempo se podría conseguir mejorar los resultados de compresión modificando y analizando el dataset más profundamente, emplear otras técnicas para mejorar la compresión con FRL haciendo modificaciones en el código y optimizando el mismo para que pueda tener compatibilidad con diferentes tipos de dataset, esto sería interesante para poder tener de forma consistente un algoritmo de compresión óptimo. Las pruebas llevadas a cabo con FRL han solo han sido favorables para los casos concretos en los que se realizaron pruebas con el algoritmo, la modificación del código es compleja dado que no sigue unas buenas prácticas de programación y hace muy difícil poder realizar modificaciones sobre el código. El *port* a C++ de FRL puede sugerir mejoras de futuro si lo que se desea es el funcionamiento tanto de la compresión como de la descompresión de datasets en nubes de puntos.

AGRADEDCIMIENTOS

Me gustaría expresar mi más sincero agradecimiento a mis compañeros Ismael Crespo Sagrado, Ashwin Kumar Gururajan, Sergi Cantón Simó y Joan Serra Sagristà por su valiosa contribución y apoyo en el trabajo sobre Point Data Cloud Compression.

A Ashwin Kumar Gururajan, quiero agradecer su dedicación en la realización de modificaciones en el algoritmo FRL, su esfuerzo por lograr su correcto funcionamiento y por compartir sus conocimientos, datos y artículos relacionados sobre su OpenTopography y resultados de compresión de FRL.

A Sergi Cantón Simó, le agradezco su generosidad al compartir su desarrollo para visualizar las especificaciones sobre los Point Data Cloud y compartir el dataset Shapenet procesado y los resultados de compresión con SparcePCC del mismo.

A Ismael Crespo Sagrado, agradezco su implicación y su disposición a compartir sus conocimientos sobre compresión y compartir el dataset S3DS procesado y con dichos resultados de compresión con TMC13.

Y a Joan Serra Sagristà, quiero expresar mi profundo agradecimiento por su tutoría, su orientación en la definición de los objetivos y su constante seguimiento y apoyo a lo largo de todo el trabajo. Su compromiso y dedicación han sido fundamentales para este proyecto.

REFERÈNCIES

[1] H. Liu, H. Yuan, Q. Liu, J. Hou, and J. Liu, "A comprehensive study and comparison of core technologies for MPEG 3-d point cloud compression," IEEE Transactions on Broadcasting, vol. 66, no. 3, pp. 701–717, 2019.

[2] R. L. de Queiroz and P. A. Chou, "Motion-compensated compression of dynamic voxelized point clouds," IEEE Transactions on Image Processing, vol. 26, no. 8, pp. 3886–3895, 2017.

[3] The KITTI Vision Benchmark Suite. (s.f.). Andreas Geiger. https://www.cvlibs.net/datasets/kitti/raw_data.php?type=person

[4] Tzamarias, D. E. O., Chow, K., Blanes, I., Serra-Sagrista, J. (2022). Fast Run-Length Compression of Point Cloud Geometry. IEEE Transactions on Image Processing, 1. https://doi.org/10.1109/tip.2022.3185541

[5] DEUTSCH, Peter. GZIP file format specification version 4.3. 1996.

[6] ISENBURG, Martin. LASzip: lossless compression of LiDAR data. Photogrammetric engineering and remote sensing, 2013, vol. 79, no 2, p. 209-217.

[7] TANG, Pingbo, et al. Automatic reconstruction of as-built building information models from laser-scanned point clouds: A review of related techniques. Automation in construction, 2010, vol. 19, no 7, p. 829-843.

[8] S. Schwarz, M. Preda, V. Baroncini, M. Budagavi, P. Cesar, P. A. Chou, R. A. Cohen, M. Krivokuca, S. Lasserre, Z. Li ´ et al., "Emerging MPEG standards for point cloud compression," IEEE Journal on Emerging and Selected Topics in Circuits and Systems, vol. 9, no. 1, pp. 133–148, 2018.

[9] Kyriazis, Ioannis Fudos, Ioannis. (2008). Identifying features by slicing point clouds.

[10] OSWAL, Savan; SINGH, Anjali; KUMARI, Kirthi. Deflate compression algorithm. International Journal of Engineering Research and General Science, 2016, vol. 4, no 1, p. 430-436.

[11] Policriti, Alberto, and Nicola Prezza. LLZ77 computation based on the run-length encoded BWT."Algorithmica 80.7 (2018): 1986-2011.

[12] Moffat, Alistair. "Huffman coding."ACM Computing Surveys (CSUR) 52.4 (2019): 1-35.

[13] Wang, Ruisheng, Jiju Peethambaran, and Dong Chen. LLidar point clouds to 3-D urban models : A review.ÏEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 11.2 (2018): 606-627.

[14] Zhang, Cha, Dinei Florencio, and Charles Loop. "Point cloud attribute compression with graph transform."2014 IEEE International Conference on Image Processing (ICIP). IEEE, 2014.

[15] SCHNABEL, Ruwen; KLEIN, Reinhard. Octree-based Point-Cloud Compression. PBG@ SIGGRAPH, 2006, vol. 3.

[16] S. Zhang, W. Zhang, F. Yang and J. Huo, "A 3D Haar Wavelet Transform for Point Cloud Attribute Compression Based on Local Surface Analysis,"2019 Picture Coding Symposium (PCS), Ningbo, China, 2019, pp. 1-5, doi: 10.1109/PCS48520.2019.8954557.

[17] HUANG, Tianxin; LIU, Yong. 3d point cloud geometry compression on deep learning. En Proceedings of the 27th ACM international conference on multimedia. 2019. p. 890-898.

[18] KAMMERL, Julius, et al. Real-time compression of point cloud streams. En 2012 IEEE international conference on robotics and automation. IEEE, 2012. p. 778-785.

[19] I. Armeni et al., "3D semantic parsing of large-scale indoor spaces," CVF Open Access, https://openaccess.thecvf.com/content_cvpr_2016/html/Armeni_3D_Semantic_Parsing_CVPR_2016_paper.html (accessed May 27, 2023).

[20] MITRA, Niloy J., et al. Registration of point cloud data from a geometric optimization perspective. En Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing. 2004. p. 22-31.

[21] "Find topography data," OpenTopography, https://portal.opentopography.org/datasets (accessed May 27, 2023).