Point Cloud Data Compression

Ismael Crespo

Resumen– En este escrito se realizará un amplio estudio del tipo de dato conocido como Pointcloud, así como de sus métodos de compresión principales, dándole una especial importancia al método FRL. Adicionalmente, Se describirán pasos previos necesarios a la ejecución de los compresores. Una comparación entre algoritmos diferentes será expuesta para entender mejor el contexto actual. Para ello se generán numerosos datos estadísticos y se revisará exhaustivamente el estado del arte actual. Además, se presentarán técnicas que mejoran el rendimiento de FRL sin incurrir una pérdida de datos. Estas técnicas están basadas en aumentar la densidad de los pointclouds utilizando diferentes técnicas.

Paraules clau– Nube de puntos, Compresión de información, FRL, Ordenación adaptativa, Fusión de Nubes de puntos, Depuración de Nubes de puntos

Abstract– In this paper a wide study of the data type known as pointcloud will be made, as well as its main compression methods, giving special importance to the FRL method. In addition, previous steps necessary to the execution of the compressors will be described. A comparison between different algorithms will be exposed to better understand the current context. By generating numerous static data and reviewing the current state of the art. Techniques that improve FRL performance without incurring data loss will be presented. Based on increasing the density of the pointclouds using different techniques.

Keywords– Point Cloud, Information Compression, FRL, Adaptive Sorting, Point Cloud Merging, Point Cloud Debugging

1 Introducción - Contexto del trabajo

No es ningún secreto que la compresión de información como disciplina ha sido, es y seguirá siendo de vital importancia, debido a que cimienta las bases de la viabilidad de cualquier actividad, científica o no, cuyo foco esté en las tecnologías de la información. Sin ir más lejos, sin compresión de información, el simple envío de datos utilizando cualquier tipo de red, se convertiría en un reto prácticamente insuperable para la infraestructura subyacente.

Con el auge de tecnologías como la realidad aumentada o virtual en los últimos años, se ha identificado la necesidad de comprimir un nuevo tipo de datos, los modelos tridimensionales. Existen varias formas de representar este tipo de datos, entre ellas se encuentran los point clouds, o nubes de puntos, estos gozan de gran popularidad debido a su simplicidad y a su precisión. Varios algoritmos actuales logran el cometido de reducir el tamaño de estas nubes de puntos, pero debido al gran volumen de datos de este tipo que existe, se intentan superar los estándares día a día.

Para poner este hecho en perspectiva, una de las tecnologías en auge que tiene a los point clouds como núcleo es Lidar (Light Detection And Ranging). Lidar es un sistema de medición masiva de posiciones, basado en un sensor de barrido láser que emite pulsos y registra los retornos una vez esos pulsos rebotan contra la superficie. Este sistema genera nubes de puntos muy pesadas, que deben ser comprimidas. Sobretodo teniendo en cuenta que se está comenzando a utilizar esta tecnología en aplicaciones que requieren de una gran interactividad, como pudieran coches autónomos o cámaras inteligentes. Comprimiendo correctamente, se minimiza la cantidad de información que se debe transmitir, disminuyendo el tiempo de envío y mejorando el tiempo de respuesta.

A lo largo de este escrito, se compararán una serie de métodos de compresión de point clouds utilizados actualmente, de entre ellos, se le dará una mayor importancia a FRL, que se describirá en profundidad junto con los otros algoritmos en una sección posterior. Se expondrán resultados prácticos de cada uno de los métodos, cuya discusión ayudarán a mejorar los rendimientos actuales y a obtener una versión global actual sobre los problemas

• E-mail de contacto: ismael.crespo.sagrado@gmail.com
• Mención realizada: Tecnologias de la Informació
• Trabajo tutorizado por: Joan Serra Sagristà (DEIC)
• Curso 2022/2023

de este tipo de datos, así como de las aproximaciones que se han descrito en la literatura para lidiar con dichos inconvenientes.

Esta investigación se realizará, hasta cierto punto, de manera colaborativa con los siguientes compañeros:

- Sergi Cantón: Estudiante de Matemática computacional en la UAB
- Xavier López: Estudiante de Ingeniería informática en la UAB
- Ashwin Kumar: Estudiante de Máster en la UAB

Pese a que la investigación de cada uno es independiente, la enorme similitud entre los campos de estudio ha hecho que la colaboración no solo haya sido provechosa para todos los miembros del equipo, sino que se decidió integrar esta cooperación a la metodología a seguir, como se discutirá posteriormente.

2 OBJETIVOS

El principal objetivo de este trabajo consiste en un estudio amplio de las tecnologías actuales de compresión de nubes de puntos tridimensionales. Se estudiarán: algoritmos, campos de uso, debilidades y puntos fuertes. Así mismo, se realizarán comparativas de un mismo conjunto de datos para los diferentes algoritmos con el objetivo de identificar los aspectos mencionados anteriormente.

Uno de los algoritmos al que más peso se dará a lo largo de este escrito es a FRL (Fast Run-Length Compression) [1]. Este nobel algoritmo de compresión de nubes de puntos sin perdida ha sido desarrollado y promovido por la Universidad Autónoma de Barcelona. Y presenta mucho potencial frente a los estándares actuales, tal y como se puede ver en la literatura actual. Es por ello que se ha realizado un esfuerzo mayor no solo en la investigación de este método concreto, sino también en posibles mejoras de su implementación.

Cabe destacar que, para el propósito de esta investigación, se centrarán los esfuerzos en sistemas de compresión de la información geométrica de las nubes de puntos. Esto es debido a que es la información más pesada de la estructura y también es aquella cuya representación es menos variables, ya que siempre consiste de tres coordenadas, a diferencia de la información semántica que pueda variar en gran medida dependiendo de los datos concretos. Además, los métodos a estudiar serán siempre sin pérdida.

Primeramente, se deberán escoger una serie de datos para comenzar la investigación. Con ellos se realizarán comparativas entre los diferentes algoritmos partiendo de un input común. Cabe destacar que un proceso de depuración de estos datos para adecuarlos a cada algoritmo será probablemente necesario. Una vez el proceso anterior finalice, será necesario entender estos datos para comprender que está ocurriendo dentro del algoritmo. Por consiguiente, calcular parámetros estadísticos será primordial. Con ellos, se podrán clasificar los datos en función de sus características geométricas concretas, lo que permitirá predecir comportamientos y detectar anomalías.

A continuación, se realizarán pruebas utilizando diferentes métodos de compresión, los cuales se describirán a posteriori, se enfocará este proceso en comparar el estado del arte actual con FRL. Para comprender y detectar comportamientos anómalos que puedan ser discutidos y permitan proponer cambios concretos, con el objetivo de mejorar la tasa de compresión de FRL. Adicionalmente, esta comparación ampliará lo actualmente establecido para FRL, aportando más información y verificando la existente.

Con toda esta información, como se ha dicho anteriormente, idear un proceso que mejore el rendimiento de FRL será el siguiente objetivo. Este puede ser aplicado antes o durante la ejecución del propio compresor. Pero una mejora notable es más que plausible si se identifican bien las debilidades del mismo. Los procesos de mejora de FRL que no sean fructuosos, no se catalogarán como inútiles, ya que la información de un posible fracaso podrá servir para futuras investigaciones.

Por último, se expondrán una serie de conclusiones que concentren toda la información adquirida a lo largo de la investigación para que futuras aproximaciones puedan sacar provecho de lo ya estudiado sin necesidad de tener que probarlo todo de nuevo.

3 ESTADO DEL ARTE

3.1 Point Clouds

Como para toda compresión de información, entender el tipo de datos que se comprimen es clave para determinar cuáles son las características concretas a las que se deben dar más o menos peso. Es por ello que una buena definición de los Point clouds es un buen punto de partida. Un point cloud es un tipo de dato consistente en una serie de puntos tridimensionales, es decir, de tres coordenadas (X, Y, Z). A esta información se le conoce como información geométrica, ya que ubica cada uno de los puntos en el espacio. Adicionalmente, un point cloud puede contar con parámetros adicionales que proporcione más información de cada uno de los puntos, a esto se le denomina información semántica. Estos parámetros adicionales pueden ser de muchos tipos dependiendo de las necesidades del modelo a representar. Algunos ejemplos pudieran ser los siguientes:

- Clasificar objetos: Cada punto puede estar ligado a un parámetro que lo clasifique dentro de una clase o una categoría específica
- Información del Color: Este es el atributo más común y, como su nombre indica, proporciona información sobre el color de cada uno de los puntos, normalmente en formato RGB.
- Datos físicos: Atributos físicos como la temperatura, la densidad o la distancia pueden llegar a utilizarse en casos concretos.

Una vez visto lo anterior, resta preguntar para qué son utilizados los point clouds en la actualidad. La aplicación más empleada actualmente va de la mano con la creciente tendencia actual en lo referente a las tecnologías de realidad virtual. En la figura 1 se puede observar un ejemplo de la información geométrica de una nube de puntos que represen-

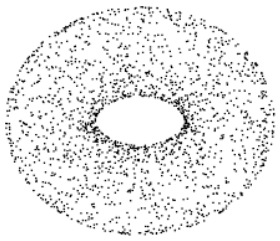

Fig. 1: Representación de la informacíon Geométrica de un Point Cloud

taría una rosquilla, que pudiéramos encontrar en cualquier entorno virtual.

No obstante, los point clouds no son el único tipo de dato que se utiliza en este contexto, existen también las meshes. Estas son otra forma de representación basada en aproximar la geometría del objeto utilizando triángulos en la superficie del mismo. La principal ventaja de los pointclouds frente a esta alternativa es que son notoriamente más precisos. Es por ello que son tan utilizados. Esta característica es también una de las razones de por qué es importante comprimirlos, para que el tamaño que implica su precisión no sea tan determinante. En adición a lo anterior, los pointclouds son utilizados en otros campos, como pudieran ser las técnicas de escaneo de objetos como pueden ser la fotogrametría o el escaneo láser terrestre (LIDAR), debido, de nuevo, a su precisión y a su consecuente facilidad de análisis.

3.2 FRL

FRL, es un algoritmo de compresión de la información geométrica de una nube de puntos sin perdida. El término sin pérdida implica que el archivo resultante al realizar el proceso de descompresión es siempre igual a los datos originales introducidos. Es importante realizar esta distinción a la hora de realizar comparativas, ya que, por norma general, un algoritmo con perdida comprimirá mucho mejor que uno sin pérdida. Ahora bien, existe la desventaja de que parte de la información se pierde en el proceso de descompresión.

Cabe destacar que este algoritmo trabaja en un espacio voxelizado, por consecuente, es necesario entender que es un voxel. Un voxel es un cubo de 1x1x1 unidades cúbicas, siendo esta unidad la que se desee, que tiene dos posibles estados: Ocupado y vacío. Un punto corresponde a un vóxel ocupado, mientras que un vóxel vacío corresponde a espacio sin puntos.

Para explicar este algoritmo en detalle se dividirá la explicación en tres fases principales:

3.2.1 Proceso de extracción

Es este primer paso se organiza el point cloud tridimensional en rebanadas o slices bidimensionales, dependiendo de la ordenación del algoritmo, por defecto, las slices se generan a partir de la condenada Z, aunque esto puede ser modificable.

En este proceso, se recorre cada rebanada bidimensional utilizando una técnica denominada raster-scan. Cada recorrido o run finaliza cuando se encuentra un voxel ocupado, en este momento, se almacenan las coordenadas de inicio y finalización de este recorrido concreto. De este modo, se sabe que todo voxel que está comprendido entre runs está vacío, siendo los únicos voxeles ocupados aquellos que figuran como límites de un recorrido concreto. Por tanto, en cada slice se realiza un conjunto de runs, si esta run no ha finalizado y se ha acabado la slice, el recorrido finalizará en el último voxel de susodicha slice independientemente del estado de ocupación del mismo. Si una run dura toda una slice, es decir, no se detecta ningún voxel ocupado, esta se catalogará como slice vacía y se almacenará de forma diferente.

Para mejorar el rendimiento, se introduce el Signaling rectangle, en este rectángulo bidimensional están comprendidos todos los puntos del point cloud. Por consiguiente, las runs se hacen siguiendo los límites de este rectángulo, no del point cloud, ya que, fuera del mismo, es seguro que todos los voxeles estarán vacíos.

3.2.2 Asignación de contextos

En este proceso, se genera un diccionario de contextos para cada slice, en este diccionario, se almacena un valor numérico de contexto por cada voxel. Ahora bien, en este diccionario se eluden aquellos voxeles que no tengan contexto, es decir, que su valor sea 0. Este contexto propio de cada voxels calcula utilizando un mapa de contextos concreto que tiene en cuenta la ocupación de 5 voxeles de la slice donde se ubica el voxel en cuestión y 9 de la slice anterior.

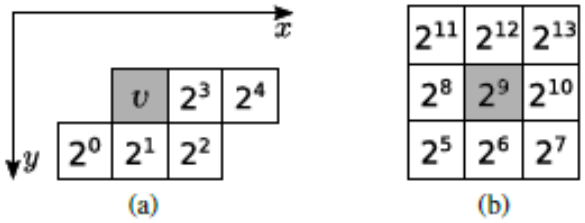

Fig. 2: Según el paper de Dion Tzamarias [1]. Mapa de contribución de contextos

Observando el mapa de contextos de la figura 2, se puede entender que cualquier estado de ocupación de los diferentes voxeles del mapa se puede codificar utilizando un valor concreto, cuyo rango será de 0 a $2^{14} - 1$

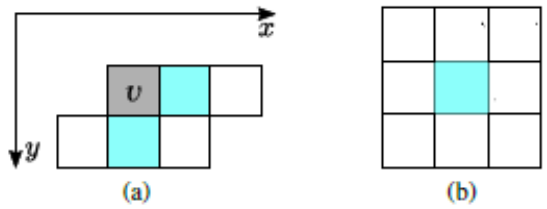

Fig. 3: Ejemplo de un cálculo del contexto del voxel v, siendo los voxeles en cian los ocupados (a) slice principal (b) slice anterior

En la figura 3 se expone un caso concreto ubicando el voxel v como voxel a estudiar. En este caso, el contexto de este voxel se podría codificar como $2^3 + 2^1 + 2^9 = 522$.

Siendo este número la inequívoca representación del contexto de este voxel. Sin necesidad de utilizar una matriz de ocupación, la cual cosa aligera los cálculos y mejora la velocidad de ejecución de forma notoria El rectángulo de señalización también se utiliza en esta fase, porque no se computan las entradas del diccionario de voxeles que están fuera del mismo.

3.2.3 Cálculo de las probabilidades y codificación aritmética

Tomando como entradas los resultados de los dos procesos anteriores, este proceso computa un conjunto de probabilidades utilizando las predicciones y el resultado de las runs, estas probabilidades serán las que se codificarán utilizando la técnica de codificación aritmética.

3.3 TMC13 - G-PCC

TMC13 es un método de compresión que utiliza el algoritmo G-PCC [4] Este está basado en una técnica denominada octree decomposition [5]. Esta técnica tiene como primer paso representar el pointcloud de manera voxelizada. Este espacio voxelizado se va dividiendo en subespacios más pequeños, siempre y cuando exista en voxel ocupado dentro de esa subdivisión. Hecho esto, se crea la estructura de Octree, esta estructura en forma de árbol indica de manera gráfica cuáles han sido las subdivisiones que se han realizado, para saber como descomprimir a posteriori. Como

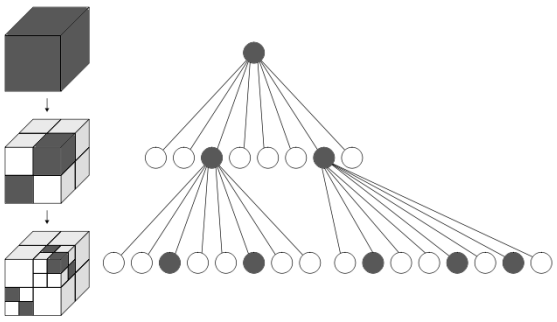

Fig. 4: Ejemplo básico de descomposición por octree

se puede observar en la figura anterior, se va dividiendo el espacio voxelizado siempre que se encuentre más de un tipo de voxel en el espacio. De este modo, si se observa el estado del cubo, en una primera instancia se ha dividido el espacio en 8. En la siguiente iteración únicamente se han vuelto a dividir 2 de esos 8 espacios. Esto implica que 6 espacios contienen voxeles de un solo tipo, ya sean ocupados o vacíos, y se pueden comprimir en conjunto sin incurrir pérdida. Para almacenar toda esta información de las divisiones se utiliza una estructura jerárquica de árbol como la que encontramos a la derecha. Una vez este proceso haya finalizado y el árbol esté en memoria, se comprime utilizando codificación aritmética.

3.4 GZIP

GZIP es un sistema de compresión clásico capaz de comprimir cualquier tipo de datos, utilizado ampliamente en envíos de información en la WWW. A pesar de no estar recomendados para pointclouds, varios proyectos que utilizan este tipo de datos utilizan también GZIP, ya sea total o parcialmente para motivos de compresión, es por eso que se ha considerado relevante. GZIP se basa en dos técnicas básicas:

- Codificación de longitud de ejecución (RLE): Este método busca secuencias de datos repetitivos y se utiliza para reducir la cantidad de información necesaria para representar una secuencia ya vista anteriormente. En RLE, una secuencia repetitiva se reemplaza por un símbolo especial seguido de la cantidad de repeticiones. Esto funciona para datos que tienden a repetirse mucho, en el caso de los pointclouds, pudieran ser coordenadas o parejas de coordenadas.

- Codificación por diccionario: RLE se combina con un diccionario de apariciones que identifica cada aparición y la mete en un diccionario, este diccionario va creciendo a lo largo del tiempo a partir de combinaciones de caracteres de entradas anteriores, substituyendo el susodicho carácter por la entrada del diccionario en cuestión.

Como se puede intuir, este método funciona bien cuando los datos son estructuralmente repetitivos, por consecuente, merece la pena probar como se comporta dado un pointcloud como input.

3.5 Métodos adicionales

3.5.1 LASZIP

LASZip [6] es un método muy especializado en un tipo concreto de pointcloud, este es el pointcloud LIDAR. Al ser especializado puede utilizar métodos con la certeza de que todos los datos de entrada seguirán un patrón, un ejemplo de esto puede ser la codificación delta. Que aprovecha la correlación entre puntos adyacentes, para, en lugar de almacenar los valores absolutos de las coordenadas, almacenar las diferencias entre los valores de punto a punto. Esto reduce aún más la cantidad de información necesaria para representar los datos. Pero solo en pointclouds cuyos puntos estén muy adyacentes, siendo LIDAR un buen ejemplo de ello. Adicionalmente, utiliza técnicas de run-length encoding, como GZIP. Aprovechando, de nuevo, la repetitividad de los datos.

3.5.2 Machine learning

Por último, comentar la existencia de algoritmos de machine learning para comprimir nubes de puntos. Estos métodos tienden a requerir tiempos de ejecución más elevados. Sobre todo teniendo en cuenta la necesidad de entrenar los modelos previamente. Se utilizan sobre todo técnicas de redes neuronales para la confección de autoencoders, estos utilizan numerosos modelos de aprendizaje para buscar la mejor representación posible a partir de unas entradas concretas, cuantas más y más variadas sean las entradas de entrenamiento, mejor será la compresión.

4 METODOLOGÍA

Para llevar a cabo esta investigación se ha seguido una metodología concreta basada en reuniones periódicas con los miembros del equipo disponibles. Debido a la multidisciplinaridad del mismo, se logró complementar los conocimientos de unos y de otros para poder llegar a conclusiones comunes. Adicionalmente, a estas reuniones, se habilitaron varios canales de información para poder compartir tanto conjeturas e inquietudes de cada uno de los miembros respecto a un tema específico de su línea de investigación, como datos útiles para los objetivos de cada uno. Por consiguiente, la colaboración y el trabajo en equipo han sido claves para poder cumplir los objetivos y obtener resultados provechosos.

Independientemente de lo anterior, debido al gran volumen de datos con los que se ha tenido que trabajar. La paralelización de tareas ha sido más que necesaria. La solución que se encontró fue intercalar tareas que requieran ejecuciones computacionalmente costosas con revisiones del estado del arte o con análisis de datos obtenidos en procesos anteriores, con el objetivo de optimizar todo lo posible el tiempo.

Para la mayoría de procesos, la metodología que se ha utilizado se podría resumir en los siguientes puntos:

- Estudio previo de los datos a comprimir, así como del método concreto para intentar anticipar los resultados
- Ejecución del método de forma ordenada y escalonada
- Confección de volumetrías a partir de los resultados obtenidos
- Observación de las volumetrías con el fin de contrastar las hipótesis iniciales.
- Estudio de posibles anomalías, ya sea por los datos o por el método.
- Estudio de posibles soluciones o mejoras para paliar las anomalías
- Discusión y exposición de los resultados con los miembros del equipo

Este proceso se ha ido repitiendo para cada una de las ramas de la investigación. Por consiguiente, se puede afirmar que se ha utilizado una metodología iterativa marcada por las reuniones periódicas y la rigurosidad a la hora de planear los siguientes pasos. Para aclarar los puntos anteriores, cabe destacar que se entiende como anomalía un comportamiento que no siga el descrito en la literatura actual. Un ejemplo de anomalía pudiera ser si para un archivo concreto, la tasa de compresión fuera anormalmente alta respecto a la afirmada en el estado del arte.

5 DESARROLLO

A lo largo de esta sección, se describirán los principales procesos que se han llevado a cabo para lograr los objetivos

5.1 Datos de prueba

La decisión de escoger un conjunto de datos de prueba que pueda satisfacer los objetivos de la investigación no es trivial. Debe ser un dataset que no se haya probado a comprimir en una investigación concreta, para aportar información nueva. Por lo que se descartan los datasets de prueba proporcionados por JPEG [3], que eran una buena opción. Se han determinado unos datos de prueba proporcionados por Stanford [2].

El conjunto de nubes de puntos de este data set, representan una serie de Áreas de oficinas que están divididas en diferentes habitaciones, a su vez, susodichas habitaciones se encuentran subdivididas en objetos. Esta serie de objetos serán los que se comprimirán. Esta clara clasificación hará bastante sencilla la navegación entre los datos para su procesamiento y estudio.

La principal razón que ha destacado este dataset frente a otros radica en la variedad en las características de las nubes de puntos. Los hay geométricamente muy complejos, como pudieran ser sillas modernas, estanterías, pilas de folios desordenados... pero también los hay sumamente simples, como los techos o los suelos, que son esencialmente objetos rectangulares tridimensionales sin contar también las diferencias de tamaño entre ellos. Estas diferencias serán de utilidad para determinar casos en los que unos algoritmos no consiguen tan buenos resultados como otros, así compara estudiar su comportamiento per se.

Otros miembros del equipo han escogido un conjunto de datos de prueba diferentes al descrito anteriormente. También se han tenido en cuenta de forma auxiliar para la discusión de resultados y pruebas concretas.

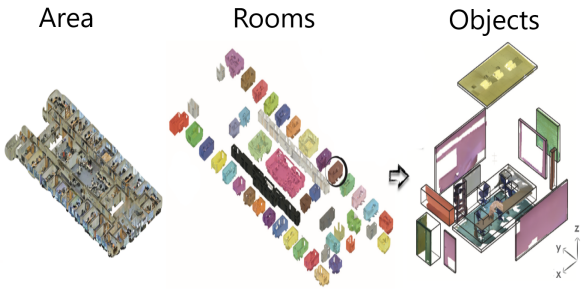

Fig. 5: Representación del Data Set y su distribución

5.2 Depuración de datos

Viendo lo anterior, cabe destacar que este dataset requiere de una depuración previa, proceso con el que no cuenta FRL de manera integrada. Por consecuente, se ha desarrollado un método de depuración que realiza los siguientes pasos:

- Eliminar los decimales redundantes.
- Convertir los puntos flotantes en enteros sin perder precisión mediante la multiplicación de una potencia de 10, siendo el exponente el número máximo de decimales redundantes a eliminar.
- Restar el valor mínimo de cada una de las coordenadas a todas las demás, eliminando los números negativos y reduciendo drásticamente el rango medio de cada una de las dimensiones.

- Eliminar duplicidades.

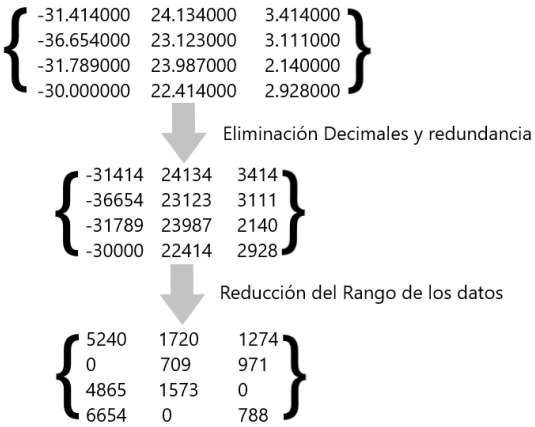

Fig. 6: Proceso de depuración de datos

Con esta depuración, se cerciora el correcto procesamiento de los datos en cualquiera de los sistemas de compresión a estudiar. Durante este proceso se han obtenido varios datos estadísticos que, aunque escasos, permiten observar características concretas de los datos, como el rango de sus valores o el tiempo requerido para su depuración. Estos datos pueden ser útiles para implementar este proceso de depuración dentro del algoritmo FRL, para el objetivo de este escrito, este proceso se ha mantenido externo a cualquiera de los algoritmos a estudiar.

5.3 Módulo Estadístico

Para poder seguir la metodología correctamente, se deben calcular parámetros estadísticos que permitan una correcta clasificación de los datos. Cabe destacar que estos parámetros han sido pensados entendiendo una imagen tridimensional como conjunto de imágenes bidimensionales, siendo la ordenación de coordenadas la cara por la que se corta este objeto tridimensional en planos bidimensionales. Esta visión se ha tomado debido a que esta es la forma en la que trabajan varios algoritmos de compresión a estudiar. Los parámetros escogidos son los siguientes:

- Rango de valores de cada una de las tres coordenadas.
- Número de slices vacías dependiendo de la ordenación de coordenadas
- Número de voxeles ocupados para cada una de las slices dependiendo de la ordenación de coordenadas. Este valor se dará en máximo, mínimo, media y mediana.
- Vértices de los rectángulos que recogen todos los puntos dependiendo de la ordenación de las coordenadas. Cabe destacar que se necesitarán dos parejas de coordenadas para cada una de las ordenaciones. A este parámetro se le conoce como rectángulo de señalización

Adicionalmente, utilizando estas estadísticas, es posible atribuir características concretas a determinados tipos de pointclouds, con el objetivo de clasificarlos y que las comparativas tengan más valor y permitan extraer conclusiones de manera clara y sencilla.

5.4 Triaje de los métodos

Los métodos a estudiar serán los descritos en el estado del arte:

- FRL: Será el algoritmo principal a estudiar
- TMC13: Rival de FRL, se utilizan para el mismo contexto y presentan tasas de compresión muy similares
- GZIP: Método de compresión muy utilizado cuya viabilidad para compresión de pointclouds está siendo discutida
- LASzip: Método especializado en comprimir un tipo de pointcloud concreto denominado (LIDAR)

También se mencionarán métodos de machine learning que, pese a diferir bastante con los métodos a estudiar, ayudan a proporcionar una visión más global de todo los métodos en circulación actualmente. Para ejecutar los métodos, se ha decidido utilizar un entorno virtualizado que utiliza el sistema operativo Ubuntu, el cual es recomendado por gran parte de los métodos de compresión a estudiar. Con toda esta información y siguiendo la metodología mencionada con anterioridad. Se formularán las hipótesis a estudiar.

5.5 Formulación de Hipótesis

Uno de los objetivos propuestos es el de mejorar el bit-rate del método FRL. Esto, sin duda, es una tarea compleja, aunque existe un concepto de suma importancia que puede ser un buen punto de partida. Como se expuso en el estado del arte, FRL comprime mejor point clouds densos. La densidad es un parámetro difícil de cuantificar en un point cloud, una aproximación plausible es utilizar una de las estadísticas calculadas, siendo esta, Número de voxeles ocupados medios por slice. Ahora bien, esta característica cuenta con tres posibles valores dependiendo de como se ordenen las coordenadas. Desde una visión geométrica, este dato sería equivalente a área media de los diferentes planos bidimensionales que conforman un objeto tridimensional. Realmente, la ordenación de las coordenadas no cambia la densidad real del objeto, únicamente varía como percibe el algoritmo ese mismo objeto.

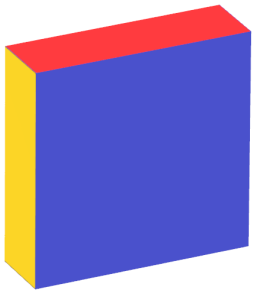

Fig. 7: En el caso de este objeto, la cara azul sería aquella con más voxeles por slice

Ya que la densidad de un point cloud no puede ser alterada juntando más los puntos entre sí, debido a que esto incurriría pérdida. La alternativa que se propone es realizar transformaciones sin pérdida que hagan, a ojos del algoritmo FRL, que el point cloud sea más densos. Un ejemplo de

estas transformaciones pudiera ser lo que se ha denominado como ordenación adaptativa:

5.5.1 Ordenación Adaptativa

La teoría es la siguiente. Si se ordenan las coordenadas de manera que la primera de ellas sea la que su estadística de voxeles ocupados medios por slice se maximiza, se debería minimizar el bit-rate. En otras palabras, se busca que el algoritmo oriente el objeto de tal manera que, de media, cuando realiza el paso de çortar"este objeto tridimensional en planos bidimensionales. La superficie sea la máxima. Según la teoría, de esta forma se maximizarían la tasa de acierto de los contextos en la predicción. La cual cosa haría que el compresor aritmético lograra comprimir de una manera más eficaz.

Esto puede no cumplirse para todos los casos, ya que, realmente, estamos extrayendo un valor medio, el cual no nos da la certeza necesaria para afirmar que los contextos predictivos vayan a ser mejores. Aun así, es una buena aproximación, por el hecho de que, aunque un mayor valor de esta estadística no implica que los contextos sean mejores, si los contextos son mejores, este valor estadístico aumentará. Por consiguiente, existe la posibilidad que este método funcione. A falta de un estudio estadístico que permita inferir la tasa de acierto del mismo.

5.5.2 Fusión de Pointclouds

Siguiendo el afán de lograr la máxima densidad posible sin incurrir una pérdida de datos. La fusión de pointclouds que originalmente estaban en archivos diferentes puede ser una opción viable, a fin de cuentas, estamos logrando más cantidad de puntos en un mismo fichero, el cual será el que servirá como input para el algoritmo FRL. La cual cosa hará, de nuevo, que los contextos sean mejores y, siguiendo con lo dicho en el apartado anterior, realizará una mejor compresión.

5.6 Ejecución de los métodos

Únicamente resta fusionar todo lo dicho en la sección para ejecutar los métodos con los datos ya depurados y analizados con el objetivo de comprobar las hipótesis sobre como mejorar FRL. En el siguiente apartado se detallarán los resultados de cada una de las partes del desarrollo.

6 Resultados

6.1 Estadísticas de los datos a comprimir

Para comenzar esta sección se mostrarán resultados estadísticos obtenidos de una serie de datos de control, se ha intentado que estos datos sean aleatorios y lo más variados posible dentro de la totalidad del data set. Como se puede observar en la Tabla 1 del anexo. Los rangos de cada una de las coordenadas de los puntos son considerablemente menores al número de puntos totales del archivo, representado en la columna 4. Esto se debe a que durante el proceso de depuración, se substrae el valor mínimo de cada una de las coordenadas al resto. La cual cosa tiene el efecto de reducir el número de bits que requiere cada condenada para ser representada, en esencia, el propio proceso depurador ya consiste una compresión.

Observando las tres últimas columnas, se puede hacer una clara distinción entre dos grupos de nubes de puntos:

- Cubiculares: Son aquellas nubes de puntos uniformes que tienden a tener un número de puntos medios por slice bastante similares para cada una de las tres coordenadas, se asemejan a cubos. Suelen ser sillas, mesas, ordenadores...
- Rectangulares: Son aquellas nubes de puntos en las que la estadística anterior es muy similar para dos coordenadas, pero difiere mucho de la tercera, en la que es considerablemente superior, Se asemejan a Rectángulos tridimensionales muy finos. Suelen ser techos, paredes, suelos...

Con esta distinción se puede garantizar que la ordenación de las coordenadas es sumamente relevante en los objetos de tipo Rectangular, mientras que puede no serlo en los objetos de tipo cubicular. Cabe destacar que esta distinción se ha podido observar no solo en los datos utilizados para las tablas, sino en la totalidad del data set. Ahora bien, no se puede extrapolar esta clasificación al resto de datasets sin realizar previamente el estudio estadístico pertinente.

6.2 Comparativa entre métodos de compresión

	FRL	TMC13	GZIP	LASzip
c_beam_1	12,80	10,37	26,79	22,95
c_beam_2	12,13	10,15	25,81	21,41
c_board_1	14,72	9,30	27,92	20,91
c_bookcase_1	15,60	11,95	30,87	32,96
c_bookcase_2	16,40	12,98	33,44	32,51
c_ceiling_1	12,77	9,21	27,90	22,50
c_chair_1	15,50	13,00	32,98	30,94
c_chair_2	15,78	13,09	33,08	32,33
c_chair_3	15,62	13,01	33,60	31,66
c_clutter_1	14,20	12,66	30,35	32,76
c_clutter_2	16,37	12,78	33,86	31,92
c_column_1	15,44	10,34	28,66	28,94
c_column_2	15,83	10,80	29,36	29,38
c_floor_1	12,74	9,54	28,67	22,77
c_table_1	13,45	11,46	30,38	25,54
c_table_2	14,19	12,34	31,76	28,05
c_wall_1	14,81	9,84	29,33	22,93
c_wall_2	15,03	9,61	28,96	22,17

Fig. 8: Comparativa entre bit-rates de distintos métodos de compresión

En la Figura 8, se puede observar una comparativa en términos de bit-rate de 4 métodos de compresión diferentes, entre ellos se encuentra FRL. El método principal a estudiar. Se puede observar una clara diferencia entre los dos primeros métodos y los dos últimos. Esto tiene sentido, ya que estos dos últimos métodos están siendo utilizados fuera de su contexto recomendado. GZIP fue concebido como método de compresión de la WWW y LASzip como

método especializado en la compresión de información de tipo Lidar. Aún y así, estos métodos siguen utilizándose para comprimir pointclouds sin contexto, la cual cosa, como se ha visto, desemboca en tasas de compresión bastante mejorables.

Ahora bien, según la literatura actual de FRL [1], este obtiene mejores resultados en a lo que bit-rate se refiere, que TMC13. Esta afirmación no se cumple en las pruebas realizadas, por consiguiente, se trata de una anomalía. Para sacar conclusiones, se requiere comparar estadística de ambos conjuntos de datos. Por un lado, los datos originales que se utilizaron en el paper original para obtener los resultados que indicaban una superioridad de FRL frente a TMC13, y, por otro lado, el conjunto de datos confeccionado para este escrito.

	S3DIS	Andrew_09
Ocup. Voxels X	23.28	778.94
Ocup. Voxels Y	231.38	479.00
Ocup. Voxels Z	950.77	764.19
Ocup. Voxels Mean	401.81	673.66

Fig. 9: Comparativa entre densidades del dataset a estudiar y el dataset estudiado en el estado del arte

En la Figura 9 se puede observar una comparativa entre los voxeles del conjunto de los datos de prueba con uno de los conjuntos del paper original: Andrew09. Estos valores apoyan la teoría de que Al no ser modelos tan densos, FRL no proporciona tan buenos resultados. Pese a que pueda parecer que no es tanta la diferencia de densidad, hay que tener en cuenta que al tratarse de valores medios, si un valor es radicalmente superior al resto hace que la estadística adultere el valor final. Esto es observable en objetos rectangulares (Tabla 2), donde una de sus tres estadísticas es hasta 90 veces más grande que las otras dos, la cual cosa adultera el valor medio del resto de los datos. Se puede afirmar, por tanto, que el dataset analizado en el paper original de FRL es importantemente más denso que el analizado en este escrito. Y esta es, sin duda, una de las razones del bajo rendimiento de FRL respecto a TMC13.

6.3 Ordenación Adaptativa

Habiendo observado las estadísticas, está claro que existen cambios en las densidades de los pointclouds dependiendo cuál es la ordenación de los mismos. Por consiguiente, solo resta probar y contrastar la teoría de la ordenación adaptativa con datos.

En la Tabla 2 se puede ver, en las seis primeras columnas, el valor de bit-rate resultante de aplicar FRL utilizando una ordenación concreta de cada uno de los archivos de prueba. Se pueden observar mejoras bastante notable en nubes de puntos de tipo aparentemente rectangular, como pudieran ser techos, suelos y paredes. Si se relaciona esta información con las dos columnas de la derecha, que indican el número de voxeles ocupados medios por slice en cada una de las coordenadas. Se puede confirmar la relación que se expuso anteriormente y que la teoría indicaba. En todos los casos, si la primera coordenada de la ordenación es aquella cuyo valor estadístico es el máximo, el bit-rate será el mínimo de entre todas las posibilidades.

Ahora bien, este cambio es sobre todo notable en archivos rectangulares, por las causas anteriormente mencionadas. Aun así, para archivos de tipo cubicular, pese a, por lo general, no corresponder mejoras tan notables, no incurre peores bit-rates. Esto permite realizar esta ordenación adaptativa independientemente del tipo de datos. Ya que, en ningún caso será contraproducente. Este hecho hace que una posible implementación en el código de FRL de este método sea computacionalmente más simple.

Sobre la viabilidad de este método, en el mejor de los casos observados, se logra reducir en 3 bits el bit-rate final. Si ponemos esta cantidad en perspectiva, se logra reducir el tamaño total del fichero en 62 kB. Este es un caso extremadamente bueno, si observamos el otro espectro, es decir, el caso en el que menos se reduce esta tasa se reduce el tamaño total en 0,2 kB. Por consiguiente se puede afirmar que este método es sumamente dependiente de los datos a comprimir, en un dataset en el que no existan datos de tipo rectangular, este método no mejorará tanto como el caso análogo, en el cual se podrían alcanzar mejorías muy notables. Para poner este ejemplo en perspectiva, el conjunto de datos totales, no solo los datos totales, sino la totalidad del dataset, pesa alrededor de 3,11 GB. Aplicando este método para la totalidad de point clouds rectangulares, se ha conseguido una reducción de 333 MB. la cual cosa implica una mejoría del 9,11%

Respecto a la ordenación de las dos últimas coordenadas, no parece ser tan sencillo como ordenar de nuevo por la estadística. Una conclusión que se puede sacar es que esta ordenación depende en gran medida de la diferencia entre ambos, si la diferencia entre ellas es notable, aproximadamente del triple, la ordenación se deberá continuar poniendo la coordenada con el valor estadístico más grande primero. En el caso contrario, no se ha logrado establecer ninguna relación. En las numerosas pruebas que se han realizado, se cumple la teoría anterior sobre la diferencia de al menos el triple, pero se requieren más pruebas para poder afirmar una relación que se cumpla siempre.

6.4 Fusión de pointclouds

Dada la naturaleza de este dataset, ha resultado sencillo encontrar una forma de juntar las nubes de puntos en una misma, simplemente se han fusionado los pointclouds que corresponden a los diferentes objetos que forman una habitación y se ha comprimido esta fusión con el objetivo de observar si obtiene mejores resultados que sus versiones por separado.

	Fusioned Area Bit-rate	Separated Area Bit-rate
ConferenceRoom_1	15.61	14.60
WC_1	16.04	15.40
Office_1	15.63	15.17
Hallway_2	14.35	13.40

Fig. 10: Comparativa entre bit-rates de habitaciones Fusionadas y por separado

En la figura anterior, se comparan los bit-rates obtenidos al

comprimir con FRL una serie de habitaciones. En la primera columna se ha comprimido un solo archivo que contiene la totalidad de los puntos de la habitación, mientras que en la segunda columna se han comprimido una serie de objetos que en conjunto forman la habitación. Los puntos de estos objetos están en ficheros diferentes y se ha realizado una media entre los bit-rates de cada uno de los ficheros. Cabe destacar que el proceso de depuración se ha realizado de forma separada, es decir, se ha depurado el fichero fusionado y los ficheros de los objetos de forma independiente. Esto se ha hecho con el objetivo de que la depuración no sea causante de los cambios en la tasa de compresión.

Observando los resultados se puede concluir que una fusión de pointclouds no es algo positivo en el contexto de estos datos, esto es debido a que, al fusionar toda la habitación, se genera una gran cantidad de espacio vacío, este espacio vacío se traduce en menor densidad, en peores contextos y en peor bit-rate. Adicionalmente, la velocidad de descompresión pasa a ser un problema, ya que el tiempo de ejecución de FRL no crece linearmente cuanto más grande sean los datos a comprimir. Su tiempo de ejecución crece de manera exponencial. Por consiguiente, cuantos más archivos pequeños tengamos, más rápido se ejecutará el método, entre otros factores, gracias al lanzado de diferentes ejecuciones del método de compresión en paralelo.

Ahora bien, se podría pensar en el método opuesto a este, subdividir archivos grandes en muchos más pequeños. Viendo lo anterior pudiera parecer una buena aproximación sin demasiadas complicaciones, pero el problema es la división en sí. Lo que se ha demostrado como una realidad es que los objetos se comprimen mejor que las habitaciones enteras, pero, dada una habitación entera, debemos ser capaces de extraer los diferentes objetos para realizar una aproximación de estas características. Esto es una tarea complicada, pero necesaria. Ya que si dividimos un pointcloud sin contexto, acabaremos con muchos archivos que contienen puntos muy alejados entre sí, la cual cosa empeorará los bit-rates.

Lejos de catalogar esta línea de investigación como fracaso, se entiende que se requiere más estudio para llegar a un sistema que permita subdividir pointclouds de manera contextualizada. Este tema está siendo estudiado por los autores de TMC13, lo cual se puede ver en su repositorio público, así que mucha más investigación debe o está siendo realizada.

6.5 Signaling Rectangle Adaptativo

Para finalizar los resultados, se propondrá un cambio en el propio algoritmo de FRL para reducir el bit-rate. Después de mucha discusión, el cambio más relevante que se pudiera implementar de manera no muy costosa tiene que ver con el signaling rectangle. Como se vio en el estado del arte, una parte fundamental de los dos primeros procesos de FRL es que no se computa nada que esté fuera del rango de este rectángulo bidimensional. Ahora bien, este rectángulo es global, es decir, se calcula para la totalidad del point cloud, lo que se propone es que este rectángulo de señalización se calcule para cada una de las slices. Esto conseguirá que las runs se efectúen de manera más compacta y no se realicen cálculos de partes del pointcloud que estén esencialmente vacíos. Para almacenar esto se podría hacer un diccionario que almacenara como índice la coordenada principal por la cual se realiza la partición y dos pares de coordenadas que correspondan a los límites del rectángulo, en el código únicamente se tendrían que cambiar las comprobaciones que se realizan con el signaling rectangle y utilizar el susodicho rectángulo adaptado a la slice concreta, utilizando la coordenada como índice. Cabe destacar que para

```
SignalingRectangles[
[0,[(123,123) , (456, 456)]],
[1,[(999,333) , (10, 1)]],
[3,[(888,222) , (100, 2)]],
[4,[(777,111) , (1000, 3)]],
....
]
```

Fig. 11: Estructura aproximada de almacenamiento de rectángulos de señalización

slices vacías no se necesitaría almacenar ninguna entrada en el diccionario. Para efectuar este cambio, primeramente se deberá estudiar si el uso de memoria adicional que este método requiere compensa la mejora de bit-rate que se efectúa, si es que se efectúa. Aun así, es un buen campo de estudio en el que FRL podría mejorar.

7 Conclusiones

Para concluir, a lo largo de este escrito se ha descrito el contexto actual de la tecnología de compresión de Point clouds. Y se han cotejado una serie de resultados tanto de los datos en sí como de los diferentes procesos de compresión, que han ayudado a identificar los principales problemas y dificultades de este tipo de dato. Se ha logrado idear un método cuya implementación es bastante sencilla que permite mejorar de manera considerable el bit-rate de algunos de los pointclouds que cumplan con la característica de rectangularidad. Esta mejora se aplica únicamente a FRL, ya que otros métodos como TMC13 no cumplen las características necesarias para que esta ordenación mejore el bit-rate resultante.

El proceso de depuración descrito es fácilmente replicable y estandarizable, siendo necesario en muchos de los pointclouds que han sido capturados con sensores modernos, como pudieran ser los utilizados para el dataset de Stanford. Por consecuente, se considera también una aportación importante.

Se han descrito formas de abordar el tema de identificar la densidad de un point cloud dependiendo de ciertas estadísticas computacionalmente poco costosas de obtener. Esto se podría aplicar a pointclouds de diversas índoles para buscar similitudes o diferencias con datos de pruebas ya testeados, como pudieran ser los utilizados en este escrito, y de este modo anticipar los resultados de varios métodos de compresión para buscar nuevas anomalías que produzcan nuevas teorías que pudieran ser puestas en práctica en un futuro.

Para acabar, cabe remarcar que este escrito ha sido ideado para servir como punto de partida para futuras investigaciones de compresión de información orientadas a pointclouds. Ya que se expone no solo el estado del arte de manera resu-

mida, sino también comparativas concretas de los mismos en unos datos concretos, datos los cuales están estudiados y caracterizados. Esta información es fácilmente extrapolable a otro tipo de datos o a otros algoritmos de compresión. Teniendo este paper como apoyo o como punto de partida. Algunas de las futuras líneas de investigación pudieran ser la investigación del punto 6.5 sobre el signaling rectangle adaptativo. Como es sabido, los problemas sobre un método no se muestran hasta que su implementación comienza, es por ello que, aunque la aproximación anterior pudiera parecer simple, los problemas subyacentes pueden y de seguro aparecerán. Es por ello que una nueva línea de investigación concreta pudiera ser fructuosa. Por último, tal y como se ha visto en el estado del arte, FRL confía mucho en el mapa de contextos actual. Este mapa de contexto utiliza la slice en la que se situa y únicamente la slice anterior, la cual cosa puede incurrir malas predicciones en pointclouds que, pese a ser densos, no lo sean suficiente para tener puntos adyacentes. Es por ello que no es para nada descabellado pensar posibles mejoras en el mapa de conceptos, aumentando el rango a varias slices extra o perfilando mejor las posiciones concretas a estudiar.

8 AGRADECIMIENTOS

Me gustaría Agradecer a los miembros del equipo de investigación por el apoyo y la información brindada. A Sergi Cantón agradecer el esfuerzo principal de confeccionar el módulo estadístico de los point clouds A Xavier Lopez por los datos relativos a Gzip y LASzip que, pese a haber sido comprobados, fueron brindados caritativamente por él Una especial mención a Ashwin Kumar y a Joan Serra Sagristà por la oportunidad de participar en un proyecto de estas características y por la paciencia y tiempo invertido que fue necesario para iniciar a alguien sin experiencia en investigación a esta disciplina. El constante feedback y resolución de dudas en las reuniones periódicas ayudaron no solo a la realización de este proyecto, sino a la observación de como trabajan profesionales de la investigación tecnológica.

REFERÈNCIES

[1] D. E. Tzamarias, K. Chow, I. Blanes, and J. Serra-Sagristà, “Fast Run-Length Compression of Point Cloud Geometry,” *IEEE Transactions on Image Processing*, vol. 31, pp. 4490-4501, 2022. doi: 10.1109/TIP.2022.3185541.

[2] Large scale parsing Stanford 2D-3D-Semantics Dataset (2D-3D-S). [Online]. Available: http://buildingparser.stanford.edu/dataset.html.

[3] JPEG Pleno Database: Microsoft Voxelized Upper Bodies - A Voxelized Point Cloud Dataset Provided by Microsoft (Charles Loop, Qin Cai, Sergio Orts Escolano, and Philip A. Chou)

[4] Mammou, K., Chou, P. A., Flynn, D., Krivokuća, M., Nakagami. (enero de 2019). ISO/IEC JTC1/SC29/WG11 N18189. Marrakech, MA. [Online]. Available: https://mpeg.chiariglione.org/standards/mpeg-i/geometry-based-point-cloud-compression/g-pcc-codec-description-v2.

[5] A. Dricot and J. Ascenso, ”Hybrid Point Cloud Geometry Coding Using Planes and Octree Representation Models,”2019 Data Compression Conference (DCC), Snowbird, UT, USA, 2019, pp. 569-569, doi: 10.1109/DCC.2019.00081.

[6] Martin Isenburg, LASzip: lossless compression of LiDAR data, LAStools. [Online]. Available: http://laszip.org.

9 APÉNDICE

Tabla 1: CARACTERÍSTICAS DE LOS DATOS A ESTUDIAR

	Range x	Range y	Range z	nov	Ocup. voxels X	Ocup. voxels Y	Ocup. voxels Z
Wall_1	7371	64	3071	95861	8.443	308.524	12.003
Wall_2	7251	94	3107	165580	22.832	1742.947	53.275
Floor_1	7422	14902	112	792653	832.491	11.668	16.889
Chair_1	670	680	851	12930	19.270	18.987	15.176
Chair_2	565	581	946	8136	14.374	13.979	8.591
Ceiling_1	6971	14950	98	967451	138.762	64.708	9772.232
Board_1	3645	56	2516	84905	23.287	1489.561	33.732
Column_1	373	1073	3072	33104	88.513	30.823	10.772
Beam_1	345	2441	898	28883	83.476	11.827	32.127
Table_1	1885	859	112	15749	8.3504	18.312	139.371
Bookcase_1	800	420	657	11425	14.263	27.137	17.363
Bookcase_2	508	1134	1168	10672	20.966	9.402	9.129
Clutter_1	515	412	189	4631	8.974	11.213	24.373

Range: El rango de los valores de cada una de las coordenadas va desde el 0 hasta el número de cada fila.

Nov: Number of Ocupied Voxels, número de puntos del pointcloud

Tabla 2: FRL BIT-RATES DEPENDIENDO DE LA ORDENACIÓN DE COORDENADAS

	(X,Y,Z)	(X,Z,Y)	(Y,X,Z)	(Y,Z,X)	(Z,X,Y)	(Z,Y,X)	Ocup. vox X	Ocup. vox Y	Ocup. vox Z
Wall_1	15.17	16.11	13.49	13.53	15.87	14.62	8.443	308.524	12.003
Wall_2	13.73	15.04	11.99	11.57	14.17	12.29	22.832	1742.947	53.275
Floor_1	12.74	12.24	14.26	14.90	13.44	14.78	832.491	11.668	16.889
Chair_1	16.15	16.19	16.28	16.20	16.83	16.88	19.270	18.987	15.176
Chair_2	15.77	15.63	16.36	16.11	16.43	16.29	14.374	13.979	8.591
Ceiling_1	14.90	12.98	14.87	12.56	12.77	12.09	138.762	64.708	9772.232
Board_1	12.91	14.184	11.99	11.57	14.17	12.29	23.287	1489.561	33.732
Column_1	12.91	13.44	13.77	15.38	15.44	16.28	88.513	30.823	10.772
Beam_1	13.29	12.09	15.11	16.12	12.80	14.97	83.476	11.827	32.127
Table_1	15.12	14.56	14.90	13.76	13.45	12.80	8.3504	18.312	139.371
Bookcase_1	14.27	15.04	13.66	15.17	15.59	15.49	14.263	27.137	17.363
Bookcase_2	17.01	16.40	17.13	16.73	16.38	16.44	20.966	9.402	9.129
Clutter_1	14.73	14.28	14.51	14.30	14.18	14.08	8.974	11.213	24.373