Learning For Compression

Antoni Reus Bergas

6 de febrero de 2024

Resumen– Cada vez los usuarios manejan más cantidad de datos, y con el avance tecnológico, estos datos ocupan un mayor espacio, más específicamente en el creciente campo de la realidad virtual, se necesitan enormes cantidades de espacio para almacenar cada uno de los elementos que se desean representar. Estos elementos, almacenados en estructuras conocidas como Point Clouds o PC, necesitan ser comprimidas para mitigar el impacto en la memoria de la que dispone un dispositivo, para ello surgen técnicas clásicas como FRL, y otras como SeqNOC que hacen uso de machine learning o ML, mediante redes neuronales, para mejorar algunos aspectos de la compresión y descompresión. En este artículo se propone un cambio en el método SeqNOC para prescindir de la necesidad de entrenar la red neuronal para cada secuencia que se desea comprimir, permitiendo usar una red ya entrenada, y posteriormente, se valorará el impacto de esta modificación en el rendimiento del algoritmo.

Palabras clave– V-PCC, G-PCC, Point Clouds, Redes Neuronales, CNN, SeqNOC, sin perdida.

Abstract– Users are handling increasingly large amounts of data, and with technological advances, this data occupies more space, specifically in the growing field of virtual reality. Enormous amounts of space are needed to store each of the elements that are desired to be represented. These elements, stored in structures known as Point Clouds or PC, need to be compressed to mitigate the impact on the memory available on a device. Classical techniques such as FRL emerge for this purpose, as well as others like SeqNOC that make use of machine learning or ML through neural networks to improve certain aspects of compression and decompression. This article proposes a change in the SeqNOC method to eliminate the need to train the neural network for each sequence that needs to be compressed, allowing the use of a pre-trained network. Subsequently, the impact of this modification on the algorithm's performance will be evaluated.

Keywords– V-PCC, G-PCC, Point Clouds, Neural Networks, CNN, SeqNOC, lossless.

1 Introducción

Con el reciente auge de tecnologías como la realidad aumentada (AR) o la realidad virtual (VR), han surgido diferentes retos con relación a como crear estas realidades o entornos, principalmente en el procesamiento, la creación y el almacenaje de estos espacios o elementos virtuales de manera viable, para que luego se puedan reproducir en sistemas como móviles o gafas donde los recursos de la máquina no son ilimitados. Es en los dos últimos puntos, la creación y el almacenaje donde surgen los Point Clouds (PC) o nubes de puntos.

Estas estructuras, como su propio nombre indica, son una nube de puntos, que representan una figura o parte de ella, ya sea en 2D o 3D, y que por lo general actúan como un todo, por ejemplo en la Fig. 1 [15], podemos ver la representación de una tetera mediante PC, con una densidad baja de puntos. Cada estructura de tipo PC contiene un conjunto de puntos organizados, y las dimensiones de la estructura a la que representa.

Por otra parte, los puntos que conforman las estructuras PC, ya no son únicamente un conjunto de bits que representan valores RGB como podemos encontrar en estructuras clásicas de representación de imágenes, sino que son estructuras más complejas, las cuales necesitamos posicionar en un punto exacto de ese espacio virtual,

• E-mail de contacto: Antoni.Reus@autonoma.cat
• Mención realizada: Tecnologias de la Información
• Trabajo tutorizado por: Joan Serra Sagristà (DEIC)
• Curso 2023/24

y por lo cual necesitamos definir su posición, además de sus características visuales como el color para poder proporcionar al usuario sensación de profundidad en esos espacios o figuras virtuales.

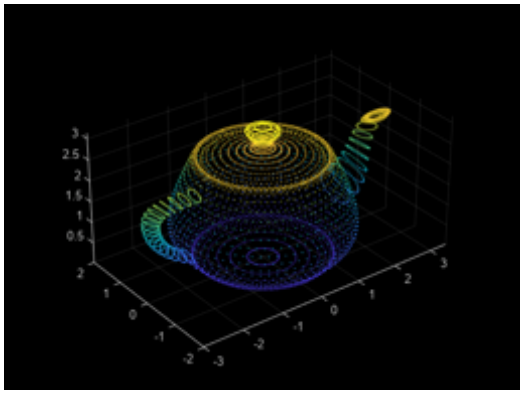

Fig. 1: Representación 3D de una taza

Estas estructuras proponen una nueva conflictiva, debido a que si deseamos crear PC con una alta definición, estos pueden estar compuestos por millones de puntos, estas nubes, se pueden almacenan en diferentes tipos de ficheros, como por ejemplo .ply o .obj, para el caso de los archivos .ply, estos son ficheros de texto, codificado en UTF-8, por lo que usa 1 Byte por carácter. Sin contar con el tamaño ocupado por la cabecera, que normalmente es despreciable debido a la gran cantidad de puntos que contienen los archivos, suponiendo una resolución de profundidad 10, cada eje x, y y z, tiene $2^{10} = 1024$ posiciones, formando un espacio 3D de $1024^3 = 1073741824$ posibles puntos. Para representar cada uno de los puentos, en el fichero .ply, necesitamos entre 1 y 4 caracteres UTF-8 para cada coordenada del punto en los tres ejes, 1 caracter para coordenadas entre [0,9], 2 caracteres para coordenadas entre [10,99], 3 para coordenadas entre [100,999] y 4 para las posiciones entre [1000,1024], siendo el caso más común el de 3 caracteres, por lo que tomaremos este caso para el ejemplo, haciendo que solo para almacenar las coordenadas del punto en el espacio 3D, necesitemos 3 caracteres por dimensión, más un separador entre coordenadas, haciendo un total de $3 \times 3 + 1 \times 3 = 12$ caracteres, además a estos 12 caracteres necesitamos añadir los caracteres necesarios para representar los atributos del punto, en este ejemplo, supondremos que solamente contiene el color del punto representado en RGB con 8 bits de profundidad, por lo que necesitaremos entre 1 y 3 caracteres para representar cada componente de color, 1 carácter para las componentes entre [0,9], 2 para valores entre [10,99] y 3 para las componentes entre [100,255], siendo este ultimo el caso más común, y el que tomaremos para el cálculo, necesitando para representar el color 9 caracteres UTF-8 más, además de 2 separadores para separar valores, y el carácter de final de línea, haciendo un total de 24 bytes por cada punto que se desea representar. Si nuestro archivo contiene, 800000 puntos, necesitaremos $24 \times 800000 = 19{,}2$ MB, pero en el caso de tener una secuencia de 5 segundos, tomada a 30 FPS, esta nos ocuparía $19{,}2 \times 30 \times 5 = 2{,}9$ GB, pudiendo apreciar la problemática de este tipo de estructura en términos de almacenamiento.

Para ilustrar de manera más clara el problema anterior y en un escenario real, en la Tab. 1, podemos apreciar el tamaño necesario para almacenar el dataset 8iVFB v2 que se expone en la sección 4.3, donde podemos ver que cada secuencia ocupa entre 4.4 y 6.7 GB.

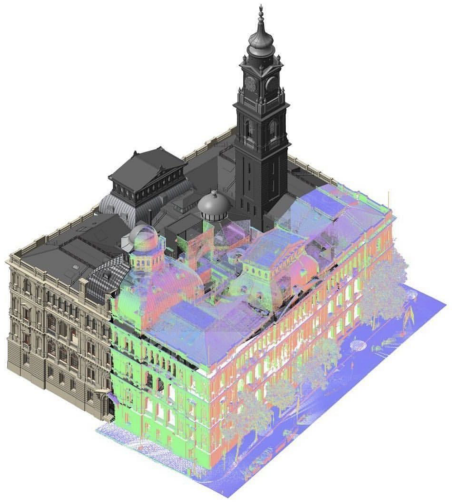

Fig. 2: Representación 3D de un edificio

Además de los casos de uso presentados al inicio como AR y VR, el uso de los PC está mucho más extendido, haciendo uso de estos en ámbitos como la robótica, escaneo 3D, la impresión de modelos 3D o en el modelado de edificios y estructuras en programas CAD, como podemos ver en la Fig. 2 [16], donde la densidad de puntos es más alta para poder representar un nivel de detalle superior.

En las siguientes secciones se expone, en primer lugar, en la sección 2, los objetivos que persigue este trabajo, en la sección 3 el estado del arte, donde se muestra la variedad de técnicas disponibles para la compresión de estructuras PC. En la sección 4 se expondrá el funcionamiento del algoritmo SeqNOC, así como se explicara las modificaciones que se le han aplicado, y por último se introducirán los conjuntos de datos que se han utilizado para valorar el impacto de esta modificación, en el rendimiento. En la sección 5 se exponen los resultados de las pruebas, y por último, en la sección 6, se discutirán las conclusiones a las que se ha llegado en este trabajo.

2 OBJETIVOS

El objetivo principal de este trabajo es la compresión de nubes de punto o point clouds, mediante el uso de técnicas de compresión que se ayudan del aprendizaje automático para mejorar los resultados. Para lograr este objetivo en primer lugar se realizará un análisis del estado actual del campo de la compresión de nubes de puntos, analizado cuáles son las técnicas más recientes en esta disciplina, mostrando las principales ramas o variantes y las ventajas que aporta cada tipo, esta parte del trabajo se llevará a cabo en la sección 3.

El segundo objetivo es profundizar en el algoritmo SeqNOC [1], mostrando el procedimiento que sigue este

	# Frames	Media Puntos	Tamaño Medio Frame	Tamaño Total
Loot	300	793834	16 MB	4.8 GB
Soldier	300	1075312	22.3 MB	6.7 GB
Long Dress	300	834328	17.7 MB	5.3 GB
Red and Black	300	727083	14.7 MB	4.4 GB

Tab. 1: 8IVFB V2: NÚMERO DE FOTOGRAMAS, MEDIA DE PUNTOS POR FOTOGRAMA, TAMAÑO POR FOTOGRAMA Y TAMAÑO TOTAL

método para lograr la compresión de dichas estructuras, y el papel que tiene el aprendizaje automático en este algoritmo, y por último se propone un cambio en el método para añadir versatilidad, de manera que además del procedimiento propuesto por los autores, que consiste en realizar un entrenamiento de la red neuronal, especifico para cada secuencia que se desea comprimir, lo que se conoce como el paradigma STM o Specifically Trained Model, previa al proceso de compresión, añadir la posibilidad de ejecutar este algoritmo siguiendo un paradigma GTM o Generally Trained Model, donde se podrá introducir al programa los parámetros de una red neuronal entrenada previamente, reduciendo de esta manera el tiempo necesario para la compresión, y permitiendo comprimir, un conjunto de PC variado con una misma red. En el objetivo de este trabajo también entra la valoración del impacto de la modificación propuesta, en el rendimiento de SeqNOC [1]. Los dos últimos objetivos se discuten en la sección 4

Para poder cumplir los principales objetivos de este documento, se han tenido que cumplir una serie de subobjetivos. El primer subobjetivo identificado es la comprensión del funcionamiento del SeqNOC [1], tanto a nivel teórico, como puede ser el procedimiento necesario para comprimir PC o secuencias de PC, o el papel que juega la CNN en el proceso de compresión, como a nivel práctico, que incluye entender el código propuesto por los autores en [2], más específicamente el uso de herramientas como Open3D o PyTorch. Un segundo subobjetivo que se ha identificado es la ejecución del código, con la documentación aportada por los autores, y la interpretación de los outputs del programa, para extraer de estos la información necesaria para la valoración de rendimiento.

Todas las pruebas realizadas han sido ejecutadas en una máquina del departamento DEIC (Departament d'Enginyeria de la Informació i de les Comunicacions), que cuenta con un procesador AMD Ryzen 5900x, 64 GB de memoria RAM y una GPU Nvidia RTX 4090, con un sistema operativo Ubuntu 22.04.3 LTS. El lenguaje que se utilizara, será Python, así como lo encontramos en la implementación del algoritmo SeqNOC [2], y todos los cambios propuestos serán publicados en un repositorio de GitHub [14].

3 ESTADO DEL ARTE

En el campo de la compresión de nubes de puntos o PCC, encontramos varias propuestas hechas por expertos, desde modelos de compresión más clásicos, como por ejemplo el FRL [4], donde se explotan las propiedades estadísticas de la información que se desea comprimir, pero para ello necesitan recorrer todos los datos, a modelos que hacen uso de técnicas más recientes, como el aprendizaje automático, como SeqNOC [1] o DGM [5], que mediante el procesamiento de un subconjunto de los datos, consiguen estimar las propiedades o características del conjunto entero, estas técnicas, permiten mejorar tiempos de procesamiento, pero también ofrecen desventajas, debido a que realmente se usan estimaciones de, por ejemplo la funcione de densidad de probabilidad, y no la función que describe la probabilidad real, haciendo que el espacio ocupado del conjunto comprimido sea mayor. Podemos ver las diferentes clasificaciones que se pueden utilizar para la compresión basada en aprendizaje en la Fig. 3 [3].

Otro dominio en el que comúnmente se clasifican las técnicas de compresión es la perdida que ofrecen los algoritmos, existiendo algoritmos lossy o con perdidas, donde si descomprimimos datos previamente comprimidos, estos serán parecidos a los originales, pero no idénticos, como por ejemplo Learned-PCGC [8] o su contrapuesto, lossless o sin perdida, donde los datos originales y los datos descomprimidos son idénticos, este es el caso de FRL

La compresión de PC también se puede clasificar en técnicas de Static PCC o en Dynamic PCC, donde la diferencia principal es que en la primera clasificación, se trabaja con un solo PC, una imagen estática, en cambio, en la segunda técnica, se trabaja con un conjunto de PC, representando una secuencia dinámica. A menudo las técnicas usadas para Static PCC son válidas para Dynamic PCC, pero las técnicas de Dynamic PCC pueden explotar características como la correlación temporal ofreciendo un mayor ratio de compresión.

Por último, otra característica que nos permite clasificar los métodos de PCC, es que características se desea comprimir, como por ejemplo la geometría, que consiste en comprimir la información necesaria para representar los puntos que forman la geometría del PC, los atributos, que se centra en reducir la cantidad de espacio necesario para representar atributos como el color o la intensidad de los puntos, como por ejemplo el método [9], y por último la compresión hibrida, que permite comprimir tanto la geometría, como los atributos, como nos ofrece el método [10].

En los últimos años, se han propuesto diferentes métodos como los métodos SeqNOC [1] donde se propone una compresión sin perdidas de datos PC dinámicos enfocada en la geometría y realizando la compresión de PC voxelizados, mediante octree o árboles octales, y usando CNNs para la estimación de las probabilidades, otro método propuesto es el FRL [4] donde se propone también un modelo sin perdidas, aunque sin uso de aprendizaje automático, con el objetivo de comprimir la geometría sobre el dominio

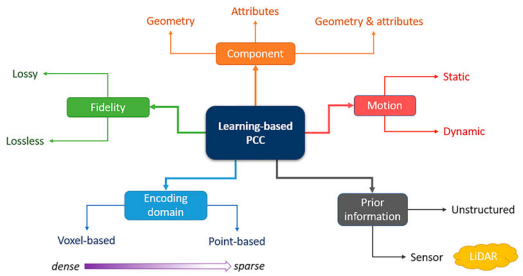

Fig. 3: Técnicas de compresión de PC basadas en aprendizaje automático

de Voxels, pero con la particularidad de no necesitar de la representación de estos en forma de octrees, haciendo un menor uso de memoria. En el mismo dominio de compresión, centrado en la compresión de geométricas y también sin perdidas, encontramos propuestas como DGM [5] o NNOC y fNNOC (fastNNOC) [6], siendo estos dos últimos (NNOC y fNNOC) los métodos predecesores del método SeqNOC, y se caracterizan por la reducción de la complejidad en la estimación de probabilidades, permitiendo la paralelización en la estimación.

En la Fig. 4 y Fig. 5, extraídas de [1], podemos ver una comparativa entre los diferentes métodos anteriormente nombrados, donde se puede ver el bitrate medio por voxel necesario y las razones de tiempos de compresión y descompresión, respectivamente, en comparación al método TMC13 [7], estándar del MPEG para G-PCC.

El resto del documento se centrará en método SeqNOC, debido al equilibrio que este nos proporciona entre el número de bits por Voxel, y el tiempo de codificación/descodificación necesarios, proporcionándonos una mejora del 25 % en el bitrate respecto del método TMC13, que nos ofrece bajos tiempos, a cambio de un alto bitrate, así como una mejora de 10.000 % en tiempos de compresión, respecto del método DMG, que ofrece un bitrate menor, pero necesita un mayor tiempo para la compresión. Como se puede apreciar la técnica SeqNOC, nos ofrece un perfil muy equilibrado comparado con las otras técnicas, además, hace uso de aprendizaje automático, adaptándose a la principal motivación del trabajo. Esta última característica, también nos ofrece un gran abanico de posibles modificaciones que se puede aplicar al método.

4 Trabajo Realizado

A continuación se expondrán, en primer lugar, una breve explicación del procedimiento que sigue la implementación de SeqNOC [1], en segundo lugar se explicara la modificación añadida, y por último que conjunto de datos se han usado para valorar el desempeño de la modificación.

4.1. SeqNOC

El algoritmo SeqNOC [1], se caracteriza por ofrecer compresión sin perdida de secuencias de PC, centrada en la geometría. En el método original, el primer paso, como se puede ver en la Fig. 6, extraída de [1], es la selección de los PC, de entre los que conforman la secuencia, que se

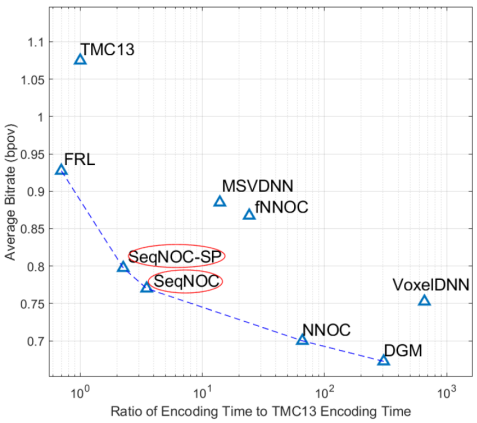

Fig. 4: Comparativa métodos de Compresión

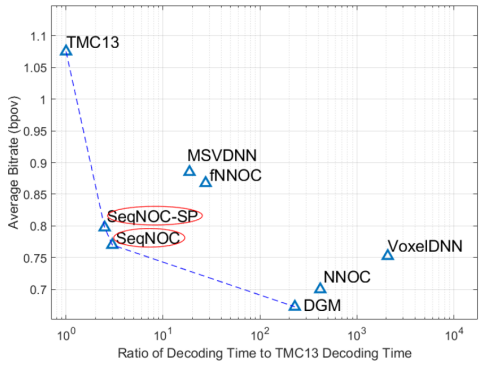

Fig. 5: Comparativa métodos de descompresión

utilizaran para entrenar la CNN, el número de PC seleccionados viene definido por los parámetros de entrada del programa. Una vez seleccionados los PC de entrenamiento, y con la estructura de la CNN se procede al entrenamiento y optimización de la CNN, para el entrenamiento la CNN recibe como entrada, segmentos de los PC de entrenamiento, obtenidos a partir de segmentación en el eje z del PC, junto a un contexto del segmento.

Una vez finalizado el entrenamiento, se procede a la compresión, dicha compresión se realiza usando un codificador aritmético, la entrada a este codificador aritmético, consiste en el segmento del PC que deseamos comprimir y un vector de 16 elementos que representa las funciones de densidad de probabilidad. El papel de la CNN es el de, a partir del segmento que se desea comprimir y un contexto, proporcionar un histograma de 16 elementos caracterizando la distribución de probabilidad de los bloques de 2×2 del segmento, estimando la probabilidad.

Por último, los resultados de la compresión se almacenan en binario, así como la red neuronal ya entrenada, que se necesitará para la descompresión del PC.

4.2. Modificaciones Aplicadas

Para cumplir los objetivos planteados en la sección 2, se han aplicado varias modificaciones al código original en [2].

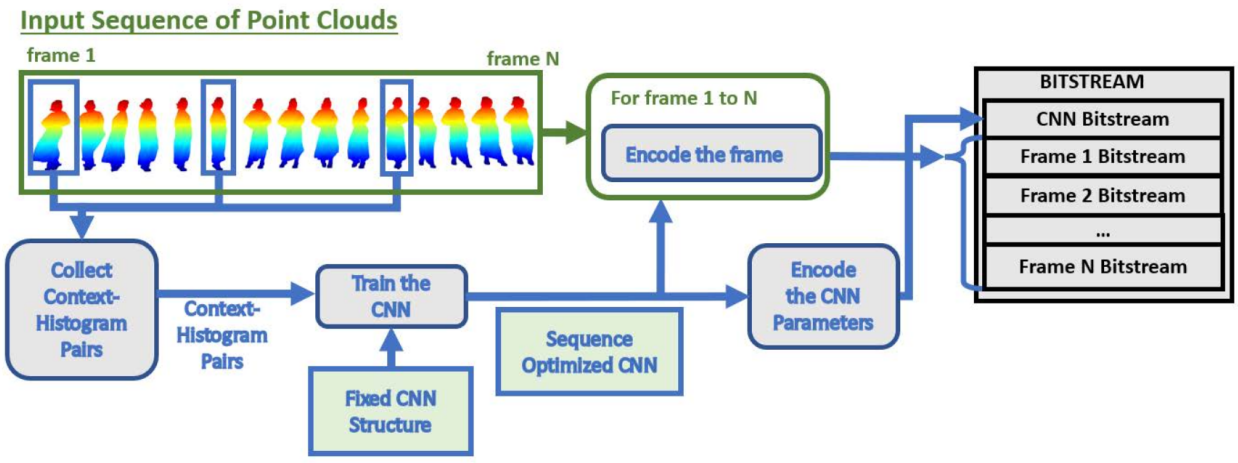

Fig. 6: Pasos para la compresión con SeqNOC

4.2.1. Solución de problemas

En esta modificación, se ha solucionado algunos errores debido a que el código publicado por los autores, estaba ideado para probar en un entorno controlado, y adaptado a los conjuntos de datos utilizados en [1], esto resultaba en errores al cambiar el entorno de ejecución o los PC de entrada, estos errores solucionados son:

En primer lugar, un error al ejecutar el método en un entorno, que no dispone de núcleos CUDA, usados por el código, para la etapa de entrenamiento de la CNN, para solucionar este problema y flexibilizar el código se ha añadido un argumento de entrada extra que permite elegir al usuario si prefiere ejecutar el código usando núcleos de la CPU o de la GPU.

El segundo problema solucionado, se producía en la etapa de optimización de la CNN, donde si los PC no tenían la misma forma que los usados por los autores, o se variaba el tamaño de los bloques usados para la optimización, producía un valor NaN en la función de optimización, que resultaba en un error al aplicar la función $forward$ para recalcular los tensores. Este problema se ha solucionado aplicando una comprobación del resultado de la función de optimización antes de ejecutar la función $forward$.

4.2.2. CNN preentrenada

Para cumplir el objetivo principal del trabajo, se ha añadido la posibilidad de evitar el entrenamiento de la CNN, proporcionando los parámetros de la red neuronal como otra entrada del programa, y segmentando el código. En este caso, el primer paso del proceso de la compresión es la estimación de distribuciones, mediante la propia CNN, para posteriormente codificar con el codificador aritmético.

Esta modificación permite varias alternativas para este método, la primera es permitir la compresión de varios PCs, entrenando una sola vez la CNN, por ejemplo, entrenándola usando un subconjunto de las nubes que se desean comprimir. Otra posibilidad es en el caso de tener un canal de comunicación entre un emisor y un receptor, si todos los PCs se comprimen con la misma CNN, solo se necesitaría transmitir una sola vez la CNN, lo que puede suponer un ahorro, en el caso de usar el canal para un gran número de transmisiones.

4.3. Datasets Utilizados

Con el objetivo de valorar como afectan estos cambios propuestos al desempeño del método SeqNOC, se van a utilizar tres datasets distintos, 8i Voxelized Full Bodies (8iVFB v2) [11], Voxelized Point Cloud Dataset for Visual Volumetric Video-based Coding (UVG-VPC) [13], y por último el Microsoft Voxelized Upper Bodies (MVUB) [12].

4.3.1. 8iVFB v2

Fig. 7: 8iVFB v2 - Soldier y Red and Black

Fig. 8: 8iVFB v2 - Loot y Long Dress

Este conjunto de secuencias, creadas por 8i Labs, y consiste en cuatro secuencias de 10 segundos obtenidos a 30 FPS, resultando en 300 nubes de puntos para cada uno de

los 4 componentes, estos componentes son cuerpos de naturaleza humana, como se puede apreciar en la Fig. 7 y Fig. 8, obtenidos mediante el escaneo con tecnología LiDAR y con una resolución de 10, lo que significa que cada vector x, y y z del espacio tridimensional tiene $2^{10} = 1024$ posiciones. La elección de este dataset se debe a que este es el usado por los autores en [1], para comparar el método propuesto con otros algoritmos.

4.3.2. UVG-VPC

Fig. 9: UVG - Elegant Dance, Casual Squat y Ready for the Winter

De manera muy similar al dataset 8iVFB v2, este dataset también se ha obtenido mediante el escaneo con LiDARs, y cuentas con una resolución de profundidad 10. En este caso, no se ha usado el dataset completo, sino que se ha seleccionado un subconjunto del dataset UVG-VPC, más concretamente, como se puede ver en la Fig. 9, se han seleccionado las secuencias llamadas Elegant Dance, Casual Squat, y Ready for the winter, para analizar que impacto tiene sobre los resultados, cuando la geometría de cada PC es más variada, ya que como podemos ver en la primera figura, la parte inferior es más ancha, en la tercera ocurre lo mismo pero con la parte superior, y en la central encontramos que la parte superior del PC, está desocupada.

4.3.3. MVUB

Fig. 10: MVUB - Andrew10, David10 y Phil10

Fig. 11: MVUB - Ricardo10 y Sarah10

Por último, se ha seleccionado el dataset MVUB, formado por cinco representaciones de la parte superior del cuerpo humano, como se puede apreciar en la Fig. 10 y Fig. 11, estas secuencias tienen una duración de 7 a 10 segundos y han sido tomadas a 30 FPS, con una resolución de profundidad 10, donde cada voxel está representado por un cubo de 0.75 mm en cada dimensión.

5 Resultados

Las pruebas realizadas se basan en observar como afecta al rendimiento del algoritmo, la modificación realizada, en términos de tiempo y espacio. Para cuantificar la compresión que nos ofrece el algoritmo, usaremos la métrica de bits necesarios para codificar cada voxel ocupado (bpov) que se describe en la ecuación 1.

$$bpov = \frac{\#\text{bits despues de compresión}}{\#\text{voxels ocupados en el PC}} \tag{1}$$

Para valorar el impacto, se ha procedido a comprimir cada una de las secuencias propuestas en la sección 4.3, con el algoritmo original, el cual entrenará la CNN para el dataset en a comprimir y comprimirá la secuencia, y en segundo lugar, se seleccionarán para cada uno de los 3 datasets propuestos, dos frames de cada secuencia, aleatoriamente, y se entrenará una CNN con los frames seleccionados, luego, esta CNN entrenada, será usada para comprimir cada una de las secuencias del dataset y se compararán los resultados. En la comparación de tiempos, aparece una suma, esta suma equivale a, en primer lugar, el tiempo requerido para la compresión de la secuencia, dividida entre el número de frames que contiene cada secuencia, el cual en todos los casos se han tomado 200 frames, y en segundo lugar, el tiempo necesitado para entrenar la CNN, dividido entre el número de secuencias. Para las pruebas GTM, donde se ha usado una misma CNN para codificar todas las secuencias, el tiempo que aparece como tiempo de entrenamiento es el tiempo necesitado para entrenar la secuencia, dividido el número de frames totales que se han codificado con esa CNN.

5.1. 8iVFB v2

Los resultados de las ejecuciones realizadas sobre el dataset 8iVFB v2 se pueden observar en la Tab 2, si observamos el impacto de la modificación, en términos de tiempo, podemos ver que la mayor diferencia la encontramos en el tiempo de entrenamiento, donde al evitar el entrenamiento para cada secuencia en el paradigma GTM, este tiempo se ve notablemente reducido respecto al paradigma STM, además este tiempo se ve reducido con cada compresión que se realice. Desde el punto de vista del espacio necesitado, podemos observar que de forma genérica, el número de bits por voxel ocupado, aumenta cuando aplicamos el paradigma GTM, las variaciones que se producen se encuentran entre el -0.06 % y el 3.21 %, en el caso de la secuencia Soldier, podemos ver como en el paradigma GTM el tamaño necesitado es menor que en el paradigma STM, pero esta diferencia es mínima.

Para valorar el impacto de usar un CNN entrenada para una secuencia de PC diferente al que se desea comprimir, se ha realizado sobre este conjunto un experimento en el

	bpov (STM)	bpov (GTM)	% bopv	Tiempo (STM)	Tiempo (GTM)	% Tiempo
Loot	0.66	0.6731	+1.98 %	3+0.6	2.9+0.175	-14.58 %
Soldier	0.7029	0.7025	-0.06 %	3.2+1	3+0.175	-24.41 %
Long Dress	0.6962	0.7107	+2.02 %	3+0.5	2.9+0.175	-12.14 %
Red and Black	0.7642	0.7887	+3.21 %	3+0.4	2.8+0.175	-12.5 %
Media	0.7058	0.7188	+1.84 %	3.05+0.625	2.9+0.175	-16.33 %

Tab. 2: 8iVFB v2 - Resultados STM vs GTM

	bpov (STM)	bpov (Soldier)	% bopv	Tiempo (STM)	Tiempo (Soldier)	% Tiempo
Loot	0.6600	0.6726	+1.91 %	3+0.6	2.9+0.33	-10.28 %
Long Dress	0.6962	0.7159	+2.83 %	3+0.5	2.9+0.33	-7.71 %
Red and Black	0.7642	0.8263	+8.17 %	3+0.4	2.8+0.33	-7.94 %
Media	0.7068	0.7383	+4.46 %	3+0.5	2.87+0.33	-8.57 %

Tab. 3: 8iVFB v2 - Resultados de la compresión usando una CNN entrenada únicamente con el PC Soldier

	bpov (STM)	bpov (GTM)	% bpov	Tiempo (STM)	Tiempo (GTM)	% Tiempo
Elegant Dance	0.5848	0.5947	+1.69 %	3+0.5	2.9+0.24	-10.3 %
Casual Squat	0.5414	0.5705	+5.38 %	2.5+0.6	2.4+0.24	-14.84 %
Ready For The Winter	0.557	0.5723	+2.75 %	2.5+1.1	2.4+0.24	-26.67 %
Media	0.5611	0.5792	+3.27 %	2.67+0.73	2.57+0.24	-17.27 %

Tab. 4: UVG - Resultados STM vs GTM

que se ha comprimido las secuencias Loot, Long Dress y Red and Black, usando una CNN entrenada con la secuencia Soldier, en la Tab. 3, se recogen los resultados de esta ejecución, donde podemos ver que en relación con la comparación entre el paradigma STM y GTM en este mismo dataset de la Tab. 2, la media de empeoramiento de bpov ha subido a un 4.4 %, principalmente por el gran aumento de esta métrica en la compresión de la secuencia Red and Black, donde vemos el empeoramiento ha sido de un 8.17 %. Este comportamiento observado, es lógico, puesto que estamos estimando las funciones de densidad de probabilidad de un conjunto de nubes de puntos, con una CNN entrenada para una secuencia diferente.

5.2. UVG-VPC

De manera similar al caso anterior, la compresión usando el paradigma GTM, proporciona un tiempo de compresión similar, pero reduce el tiempo de entrenamiento necesario, y en el dominio del almacenamiento, el paradigma GTM tiene un impacto negativo en la métrica de bpov, con variaciones de entre 1.69 % a 5.38 %, cabe destacar que las secuencias seleccionadas en este caso han sido elegidas para valorar el impacto de comprimir geometrías más variadas, y observando los resultados, vemos que el impacto no es significativo.

5.3. MVUB

Por último, al realizar las pruebas sobre el dataset de Microsoft Voxelized Upper Bodies, vemos que la diferencia en tiempo en este caso es más notable, y aparecen variaciones en las métricas de bpov de entre el 0.35 % en el mejor de los casos, hasta un 7.22 % en el caso del PC Phil10, la variación media en este conjunto es de un 4.84 %, la variación más alta de todas las pruebas realizadas, si observamos en el dominio del tiempo, la variación media que hemos obtenido es del -19.72 %, la mayor reducción de tiempo, llegando a una reducción de hasta el -26.28 % en el caso de Andrew10.

En la Fig. 12, podemos observar un gráfico donde se compara todas las variaciones porcentuales obtenidas, tanto en tamaño, como en tiempo, y las medias de cada conjunto. En general, podemos apreciar como la mejora en tiempos de ejecución, se encuentra entre el 16.33 % y el 19.72 %, con un empeoramiento en el tamaño ocupado de entre el 1.84 % en el caso del 8iVFB v2 y el 4.84 % de media para el conjunto MVUB.

6 Conclusión

El objetivo de este trabajo era el estudio de la compresión de point clouds, especialmente aquellas técnicas que hacen uso de machine learning, que ha terminado enfocándose en el método SeqNOC, proponiendo un cambio para dicho método, que le dota de más flexibilidad, y en última instancia, evaluando el rendimiento de esta modificación. Como se ha podido comprobar, esta característica añadida, ofrece tanto ventajas, como desventajas, mejorando hasta en un 19.72 % el tiempo necesario para la compresión, a cambio de aumentar como máximo un 4.84 % de media y permitiendo al usuario reducir el tiempo de entrenamiento de la CNN, tanto como se desee. La decisión de uso de esta modificación recae en el usuario final, dependiendo de las necesidades de este.

Como valoración personal, este trabajo me ha ayudado a profundizar en el campo de la compresión de datos, descubriendo otras aplicaciones de esta disciplina, así como a realizar un primer contacto con el aprendizaje automático

	bpov (STM)	bpov (GTM)	% bpov	Tiempo (STM)	Tiempo (GTM)	% Tiempo
Andrew10	0.7777	0.7804	+0.35 %	5.5+2.3	5.4+0.35	-26.28 %
David10	0.7409	0.7698	+3.9 %	7.3+2	7.2+0.35	-18.82 %
Phil10	0.7893	0.8463	+7.22 %	7.1+2.3	6.9+0.35	-22.87 %
Ricardo10	0.7379	0.7824	+6.03 %	6+1.3	5.9+0.35	-14.38 %
Sarah10	0.7421	0.7917	+6.68 %	6.8+1.5	6.6+0.35	-16.27 %
Media	0.7576	0.7941	+4.84 %	6.54+1.88	6.4+0.35	-19.72 %

Tab. 5: MVUB - RESULTADOS STM VS GTM

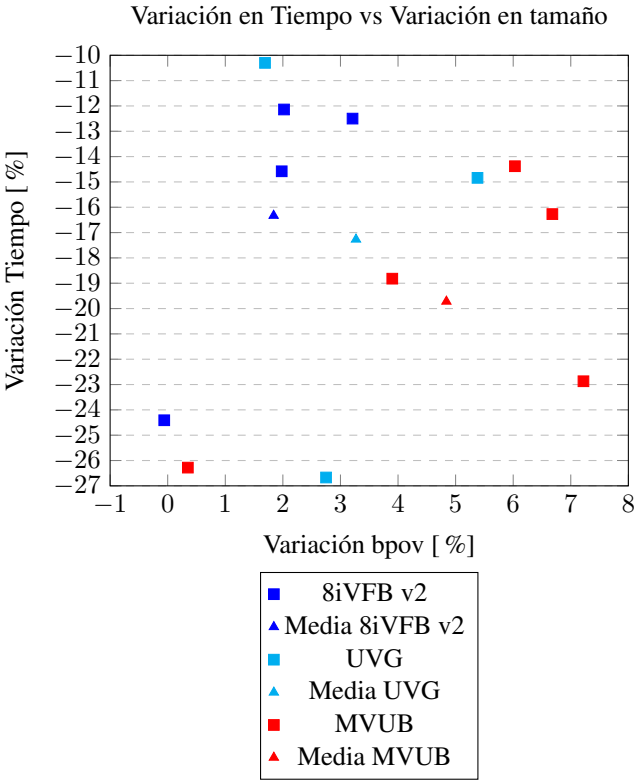

Fig. 12: Comparación de las variaciones obtenidas para cada elemento comprimido.

y descubrir las aplicaciones de esta tecnología, suponiendo muchos retos nuevos, pero que considero, han sido útiles para mi desarrollo.

AGRADECIMIENTOS

Me gustaría expresar mi agradecimiento a mi tutor, Joan Serra Sagristà por su acompañamiento a lo largo del trabajo, su ayuda en todo el proceso, y su apoyo a lo largo de este trabajo, con sus consejos y correcciones, y al DEIC por el equipo que me ha cedido para poder realizar las pruebas presentadas.

REFERENCIAS

[1] E. C. Kaya and I. Tabus, ”Lossless Compression of Point Cloud Sequences Using Sequence Optimized CNN Models,ïn IEEE Access, vol. 10, pp. 83678-83691, 2022, doi: 10.1109/ACCESS.2022.3197295.

[2] marmus12. (2022). SeqNOC, GitHub repository, https://github.com/marmus12/seqnoc

[3] Quach, M., Pang, J., Tian, D., Valenzise, G., & Dufaux, F. Survey on deep learning-based point cloud compression. Frontiers in Signal Processing, vol. 2, pp. 846972, 2022, doi: 10.3389/frsip.2022.846972

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

[5] D. T. Nguyen, M. Quach, G. Valenzise and P. Duhamel, ”Lossless Coding of Point Cloud Geometry Using a Deep Generative Model,ïn IEEE Transactions on Circuits and Systems for Video Technology, vol. 31, no. 12, pp. 4617-4629, Dec. 2021, doi: 10.1109/TCSVT.2021.3100279.

[6] E. C. Kaya and I. Tabus, ”Neural Network Modeling of Probabilities for Coding the Octree Representation of Point Clouds,”2021 IEEE 23rd International Workshop on Multimedia Signal Processing (MMSP), Tampere, Finland, 2021, pp. 1-6, doi: 10.1109/MMSP53017.2021.9733658.

[7] D. Graziosi, O. Nakagami, S. Kuma, A. Zaghetto, T. Suzuki, and A. Tabatabai, “An overview of ongoing point cloud compression standard- ization activities: Video-based (V-PCC) and geometry-based (G-PCC),”

APSIPA Trans. Signal Inf. Process., vol. 9, no. 1, pp. 1–15, 2020.

[8] J. Wang, H. Zhu, H. Liu and Z. Ma, ”Lossy Point Cloud Geometry Compression via End-to-End Learning,ïn IEEE Transactions on Circuits and Systems for Video Technology, vol. 31, no. 12, pp. 4909-4923, Dec. 2021, doi: 10.1109/TCSVT.2021.3051377.

[9] C. Zhang, D. Florêncio and C. Loop, ”Point cloud attribute compression with graph transform,”2014 IEEE International Conference on Image Processing (ICIP), Paris, France, 2014, pp. 2066-2070, doi: 10.1109/ICIP.2014.7025414.

[10] D. T. Nguyen and A. Kaup, ”Lossless Point Cloud Geometry and Attribute Compression Using a Learned Conditional Probability Model,ïn IEEE Transactions on Circuits and Systems for Video Technology, vol. 33, no. 8, pp. 4337-4348, Aug. 2023, doi: 10.1109/TCSVT.2023.3239321.

[11] Eugene d’Eon, Bob Harrison, Taos Myers, and Philip A. Chou, ”8i Voxelized Full Bodies - A Voxelized Point Cloud Dataset”, ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document WG11M40059/WG1M74006, Geneva, January 2017.

[12] C. Loop, Q. Cai, S. Orts Escolano, and P.A. Chou, “Microsoft Voxelized Upper Bodies – A Voxelized Point Cloud Dataset,” ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document m38673/M72012, Geneva, May 2016.

[13] G. Gautier, A. Mercat, L. Fréneau, M. Pitkänen and J. Vanne, ÜVG-VPC: Voxelized Point Cloud Dataset for Visual Volumetric Video-based Coding,”2023 15th International Conference on Quality of Multimedia Experience (QoMEX), Ghent, Belgium, 2023, pp. 244-247, doi: 10.1109/QoMEX58391.2023.10178589.

[14] Tritoni24. (2024). LearningForCompression, GitHub repository, https://github.com/Tritoni24/LearningForCompression

[15] The MathWorks Inc. (2022). Object for storing 3-D point cloud, Natick, Massachusetts: The MathWorks Inc. https://es.mathworks.com/help/vision/ref/pointcloud.html

[16] Measured Survey PRO. (2022). Point cloud surveys. measuredsurvey365.co.uk https://measuredsurvey365.co.uk/services/point-cloud-surveys/