

# Aplicación de curvas de llenado del espacio en nubes de puntos 3D almacenadas en Octrees

Pablo Díaz, Miguel Yermo, José C. Cabaleiro, Francisco F. Rivera

Junio 2025

# Índice

---

1. Búsquedas de vecinos
2. Curvas de llenado del espacio (SFCs)
3. Octrees
4. Resultados
5. Bibliografía

# Búsquedas de vecinos

---

Dada una nube de puntos tridimensional  $P$ , un centro  $c$  y un radio  $r$ , buscamos encontrar todos los puntos  $q \in P$  con  $\|q - c\| < r$ .

Esta operación extrae la estructura local de la nube, y tiene infinitos usos en el procesamiento de nubes de puntos: clusterización, segmentación, extracción de propiedades, ...

**Problema:** extraer los vecinos de un gran número de puntos puede ser costoso, y de hecho puede convertirse en el paso más lento dentro del procesamiento de la nube.

# Nubes LiDAR

---

- Gran tamaño, altamente irregulares en densidad y geometría
- Varios tipos (sensor aéreo vs terrestre)
- Necesidad de métodos optimizados para el cálculo de vecinos

# Curvas de llenado del espacio

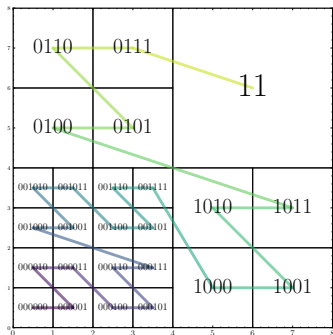
---

- Normalmente, los datos de las nubes se almacenan como un vector de puntos, con el orden de escaneo del sensor LiDAR → **Mala localidad espacial**
- Puntos muy cercanos en el espacio tridimensional se pueden encontrar muy alejados en memoria
- ¿Cómo mejorar la localidad? → Reordenando la nube mediante una curva de llenado del espacio (*Space Filling Curve*) [1]

# SFCs de Morton y Hilbert

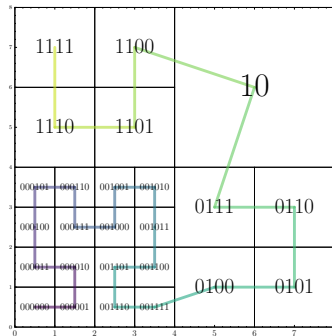
## SFC de Morton [6]

- Presenta saltos, menor localidad
- Cálculo en unas pocas instrucciones (LUTs, vectorización)



## SFC de Hilbert [4, 5]

- Continua, mejor localidad teórica (continuidad en  $\|\cdot\|_\infty$ )
- Cálculo iterativo más lento



# Reordenando la nube

---

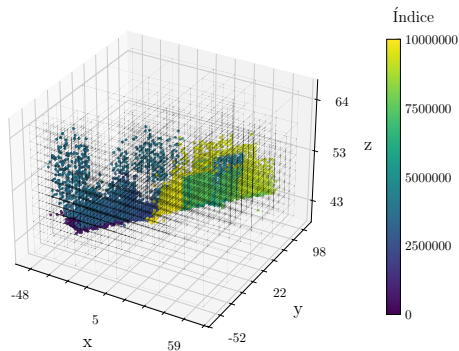
- 1 Discretización: hallar la *bounding box* y trasladar todos los puntos al espacio  $S_L = [0, 2^L) \times [0, 2^L) \times [0, 2^L) \subset \mathbb{Z}^3$ .
- 2 Para cada punto  $p = (x, y, z) \in S_L$ , hallar su código  $c$  de Morton o de Hilbert de  $3L$  bits, marcando su orden en la curva de profundidad  $L$ .
- 3 Una vez se han hallado todos los códigos, utilizarlos como índice para reordenar la nube.

# Octrees

- 1 Estructura jerárquica de volúmenes (*BVH*) donde el espacio se subdivide recursivamente en 8 octantes.
- 2 Los nodos internos tienen 8 suboctantes, las hojas tienen puntos de la nube.
- 3 Se subdivide hasta que las hojas no tienen más de  $N_{max}$  puntos.

## ¿Cómo almacenar los nodos?

- Octrees basados en punteros
- Octrees lineales



Construcción de un Octree sobre la nube Lille\_0, reordenada mediante la curva de Hilbert



# Búsqueda de vecinos en Octrees

---

## Ventajas

- Mucho mejor que fuerza bruta  $\mathcal{O}(\log N)$  en vez de  $\mathcal{O}(N)$ .
- Los puntos de las hojas están próximos en memoria gracias a las SFCs.

## Desventajas

- Se debe recorrer todo el árbol
- Muchas redirecciones de punteros, peor localidad del árbol

# Optimización con Octrees lineales

---

Nuestro Octree lineal es una variante de la estructura *cornerstone-octree* dada por Keller et al. [3], originalmente usada en simulaciones de partículas.

## Ideas principales

- Cada hoja o nodo interno se puede representar mediante el rango de índices de la nube reordenada, gracias al reordenamiento por SFCs.
- La información para enlazar el árbol y realizar las búsquedas también se puede comprimir en un array indicando el primer hijo de cada octante.

El Octree lineal cuenta con las siguientes características:

- 1 Construcción rápida y altamente paralelizable
- 2 **Rangos de índices en la rama disponible en todos los nodos, y consecutivos gracias a las SFCs**
- 3 Estructura compacta y contenida en unos pocos arrays, muy buena localidad del árbol

*Algoritmo optimizado de búsquedas*

# Setup experimental

---

Conjunto de nubes variado: *Paris-Lille*, *DALES*, *Semantic3D* y *Speulderbos*.

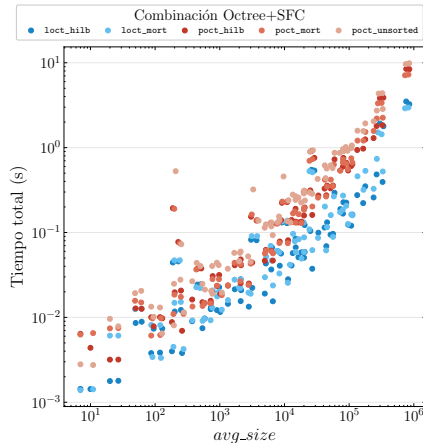
→ 2 experimentos principales:

- 1 **Búsquedas aleatorias:** 5000 búsquedas en un subconjunto de centros aleatorio  $v_c \subset P$ , con varios radios, *kernels* (esfera, círculo, cubo, cuadrado) y combinaciones SFC+Octree.
- 2 **Búsquedas completas:** lo mismo pero con  $v_c = P$ , y realizando las búsquedas con el orden de la nube tras el reordenamiento, para más localidad.

→ Paralelización a nivel de bucle con OpenMP, 40 hilos sobre arquitectura NUMA.

Octree	Reordenamiento	Nombre
Punteros	Ninguno	● poct_unsorted
	Morton	● poct_mort
	Hilbert	● poct_hilb
Lineal	Morton	● loct_mort
	Hilbert	● loct_hilb

# Resultados



Búsquedas aleatorias - Tiempo vs tamaño promedio de los vecindarios en varias nubes (log-log)

# Resultados

*Búsquedas completas -  
Varias nubes,  
kernel esférico*

Nube	$n$	$r$ (m)	Octree	SFC	Tiempo (s)
<i>Lille_0</i>	10.00M	3.0	Punteros	-	445.24
				Morton	332.11
				Hilbert	324.87
			Lineal	Morton Hilbert	130.96 <b>130.20</b>
<i>5080_54400</i>	12.22M	10.0	Punteros	-	98.04
				Morton	64.99
				Hilbert	62.80
			Lineal	Morton Hilbert	<b>42.66</b> 46.31
<i>bildstein_station1</i>	29.70M	0.1	Punteros	-	118.44
				Morton	106.35
				Hilbert	103.08
			Lineal	Morton Hilbert	<b>60.66</b> 65.11
<i>sg27_station8</i>	429.62M	0.05	Punteros	-	2549.36
				Morton	2243.24
				Hilbert	2208.13
			Lineal	Morton Hilbert	<b>1273.68</b> 1427.66

# Conclusiones

---

- TODO

# Últimos avances

---

- Método de búsqueda en Octree lineal devolviendo rangos de índices en vez de lista de punteros / índices  $\rightarrow$  Gran mejora para  $r$  grande.
- Mejorada la comprobación geométrica para kernels esféricos, a partir del trabajo de Behley et al. [2]
- Con todo esto, búsquedas mucho más rápidas ( $\approx 10x$ ) que los Octrees y KD-Trees de PCL y 2 – 3x veces más rápidas que el Octree desarrollado por Behley et al. [2]



# Referencias I

---

- [1] Tetsuo Asano et al. “Space-filling curves and their use in the design of geometric data structures”. In: *Theoretical Computer Science* 181.1 (1997), pp. 3–15.
- [2] Jens Behley, Volker Steinhage, and Armin B Cremers. “Efficient radius neighbor search in three-dimensional point clouds”. In: *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2015, pp. 3625–3630.
- [3] Sebastian Keller et al. “Cornerstone: Octree Construction Algorithms for Scalable Particle Simulations”. In: *Proceedings of the Platform for Advanced Scientific Computing Conference. PASC '23*. ACM, June 2023, 1–10. DOI: [10.1145/3592979.3593417](https://doi.org/10.1145/3592979.3593417). URL: <http://dx.doi.org/10.1145/3592979.3593417>.
- [4] Warren M Lam and JH Shapiro. “A class of fast algorithms for the Peano-Hilbert space-filling curve”. In: *Proceedings of 1st International Conference on Image Processing*. Vol. 1. IEEE. 1994, pp. 638–641.

# Referencias II

---

- [5] Yohei Miki and Masayuki Umemura. “GOTHIC: Gravitational oct-tree code accelerated by hierarchical time step controlling”. In: *New astronomy* 52 (2017), pp. 65–81.
- [6] Guy M Morton. *A computer oriented geodetic data base and a new technique in file sequencing*. IBM, Ottawa, Canada. 1966.

The slide features a central orange rectangle with the text "Preguntas?". On the left side, there is a small cluster of three squares. On the right side, there is a larger, more complex arrangement of squares, some of which are overlapping, creating a grid-like pattern.

Preguntas?