CEA Comité Español de Automática
XLVII JORNADAS DE AUTOMÁTICA CÓRDOBA 2026
UNIVERSIDADE DA CORUÑA

Jornadas de Automática

Procesamiento y visualización procedimental de nubes de puntos mediante software libre

Emiliano Pérez*, Santiago Salamanca, Pilar Merchán

Escuela de Ingenierías Industriales, Universidad de Extremadura, Avda. de Elvas s/n, 06006, Badajoz, España.

To cite this article: Pérez, E., Salamanca, S., Merchán, P. 2026. Procedural Processing and Visualization of Point Clouds Using Free and Open-Source Software. Jornadas de Automática, 47. https://doi.org/10.17979/ja-cea.2026.47.13872

Resumen

El procesamiento de nubes de puntos en el ámbito científico se ha consolidado en torno a operaciones eficientes de reducción, voxelización, análisis de vecindad y filtrado mediante bibliotecas especializadas externas a los entornos de representación 3D. Paralelamente, los ecosistemas de software libre orientados al modelado y renderizado, como Blender, ofrecen capacidades avanzadas para la visualización, iluminación, animación y composición de escenas, pero no siempre incorporan flujos accesibles para el preprocesamiento interactivo de nubes de puntos. Este trabajo presenta una herramienta procedimental desarrollada en Blender mediante *Geometry Nodes* para la reducción, voxelización, recorte espacial, filtrado de *outliers*, instanciación de primitivas y visualización de atributos asociados a nubes de puntos. El sistema se organiza como un modificador modular, no destructivo y parametrizable, complementado con un mecanismo de estimación de carga gráfica para prevenir configuraciones de visualización excesivamente costosas. La aportación principal reside en integrar, dentro de un único entorno visual enormemente popular, operaciones básicas de preprocesamiento y representación gráfica, facilitando tanto la exploración preliminar de los datos como la generación de imágenes renderizadas, animaciones y escenas enriquecidas con otros modelos 3D.

Palabras clave: Percepción y Sensorización, Fusión de información y sensores, Procesamiento de imágenes, Trabajo en entornos reales y virtuales, Interfaces inteligentes.

Procedural Processing and Visualization of Point Clouds Using Free and Open-Source Software

Abstract

Point cloud processing in the scientific domain has become established around efficient operations for reduction, voxelization, neighborhood analysis, and filtering, typically implemented through specialized libraries external to 3D representation environments. In parallel, free and open-source software ecosystems aimed at modeling and rendering, such as Blender, provide advanced capabilities for visualization, lighting, animation, and scene composition, but they do not always include accessible workflows for the interactive preprocessing of point clouds. This paper presents a procedural tool developed in Blender using Geometry Nodes for reduction, voxelization, spatial clipping, outlier filtering, primitive instancing, and visualization of attributes associated with point clouds. The system is organized as a modular, non-destructive, and parameterizable modifier, complemented by a graphical-load estimation mechanism intended to prevent excessively costly visualization configurations. The main contribution lies in integrating basic preprocessing and graphical representation operations within a single, highly popular visual environment, thereby facilitating both preliminary data exploration and the generation of rendered images, animations, and scenes enriched with other 3D models.

Keywords: Perception and sensing, Information and sensor fusion, Image processing, Work in real and virtual environments, Intelligent interfaces.

*Autor para correspondencia: emilianoph@unex.es

1. Introducción

Las nubes de puntos constituyen una forma habitual de representar información tridimensional obtenida mediante escaneado 3D, sensores LiDAR, fotogrametría, etc. A diferencia de las mallas poligonales, describen la geometría como un conjunto de coordenadas, que pueden incorporar atributos adicionales como color, normales, intensidad o campos escalares (p. ej. Temperatura). Esta representación es directa y flexible, pero tiene alguna desventaja, entre ellas cuando se desea inspeccionar, interpretar o comunicar visualmente la información, geométrica o no, contenida en ellas.

El tratamiento de nubes de puntos cuenta con herramientas muy consolidadas. Bibliotecas como Point Cloud Library (Rusu, 2011) u Open3D (Zhou, 2018) proporcionan algoritmos robustos para filtrado, registro y segmentación, mientras que aplicaciones como CloudCompare (Girardeau-Montaut, 2016) o MeshLab (Cignoni, 2008) permiten inspeccionar y procesar datos tridimensionales mediante flujos interactivos orientados al análisis técnico. Estas soluciones son muy potentes para tareas de procesado, medición o reconstrucción, pero no siempre están orientadas a la generación ágil de imágenes renderizadas, animaciones o escenas compuestas con otros elementos 3D.

En paralelo, Blender (Blender Foundation, 2025) se ha consolidado como un entorno libre de modelado, animación y renderizado 3D. Aunque su propósito original no fue el procesamiento geométrico de nubes de puntos, ofrece algunas herramientas para su visualización simple. Por otro lado, su sistema *Geometry Nodes* permite construir modificadores procedimentales no destructivos mediante grafos de nodos, lo que facilita la generación, transformación e instanciación de geometría en tiempo real dentro de la propia escena.

Existen complementos de la comunidad para la importación y visualización de nubes de puntos en Blender, como Point Cloud Visualizer (Uhlík, 2017), Blender Photogrammetry Importer (Bullinger, 2021), o flujos basados en Sverchok (Nortikin, 2013) y Animation Nodes (Lucke, 2015). Sin embargo, estas soluciones suelen centrarse en la importación, visualización o manipulación generalista de datos, dependientes de complementos externos. Además, ofrecen un flujo de trabajo editable, interactivo y transparente, basado en funciones básicas —muestreo, filtrado, etc.— aplicadas sobre los datos 3D.

Este trabajo propone una herramienta procedimental basada en Blender y *Geometry Nodes* (Gumster, 2022) orientada a la exploración interactiva, el preprocesamiento rápido y la representación simplificada de nubes de puntos. La propuesta, no pretende sustituir a bibliotecas analíticas especializadas, sino ofrecer un entorno visual no destructivo en el que distintas operaciones —reducción de densidad, voxelización, recorte espacial mediante volúmenes de control, filtrado de *outliers*, coloreado por atributos e instanciación de primitivas— puedan combinarse de forma modular. De este modo, la herramienta facilita tanto la exploración preliminar de los datos como la generación de imágenes renderizadas, animaciones y escenas 3D enriquecidas. Además, el sistema incorpora una estimación de la carga gráfica asociada a la visualización, lo que ayuda al usuario a ajustar los parámetros de trabajo y evitar configuraciones que bloqueen el PC.

El resto del artículo se organiza como sigue: la Sección 2 describe el diseño general de la herramienta, detallando la arquitectura modular de sus bloques funcionales y su lógica de implementación, la Sección 3 presenta el flujo de trabajo interactivo y los resultados demostrativos y, por último, la Sección 4 recoge la discusión, conclusiones y las futuras líneas de desarrollo.

2. Diseño general de la herramienta

La herramienta que se presenta en este trabajo se implementa como un modificador procedimental desarrollado con *Geometry Nodes* de Blender. Estructuralmente, el núcleo de la herramienta se organiza en torno a un grupo principal, denominado *VisualizadorPuntos*, que recibe la nube de puntos cargada en la escena y expone al usuario los parámetros de control del flujo de procesamiento. Desde este grupo principal, la geometría de entrada y sus atributos asociados se distribuyen hacia diferentes módulos funcionales, que pueden activarse o ajustarse de forma independiente sin modificar la nube original.

La herramienta se distribuye en un archivo en formato `.blend` (se puede descargar en la dirección web https://github.com/3dcovim/VisualizadorPuntos) que contiene el modificador de *Geometry Nodes* reutilizable, acompañado de un script auxiliar en Python y una nube de puntos de prueba. La Figura 1 resume el flujo general de funcionamiento del modificador desarrollado. A partir de una nube de puntos importada en formato `.ply`, el modificador permite aplicar una secuencia de operaciones parametrizables que pueden activarse o ajustarse desde la interfaz de usuario. El flujo principal comprende la reducción de puntos, la voxelización, el filtrado de *outliers*, la asignación de color, el recorte espacial, la instanciación de primitivas geométricas y la aplicación de materiales. La salida del sistema puede emplearse directamente para la visualización y renderizado dentro de Blender, o exportarse como una nueva geometría procesada.

A partir de esta arquitectura general, las siguientes subsecciones describen los principales bloques funcionales de la herramienta: la entrada y reducción de la nube de puntos, la voxelización, la instanciación de primitivas, el recorte espacial, el tratamiento del color y el control dinámico de la carga visual.

2.1. Reducción y preparación de la nube de puntos

La primera etapa del flujo corresponde a la reducción de la nube de puntos. Para ello, el submódulo de diezmado implementa las dos siguientes estrategias de muestreo:

- **Reducción porcentual:** Permite retener una fracción aleatoria o indexada de la nube en función de un parámetro escalar de control $r \in [0, 100]$. Si la nube original consta de N_0 puntos, el número aproximado de elementos resultantes N viene determinado por:

$$N \simeq r \cdot \frac{N_0}{100} \tag{1}$$

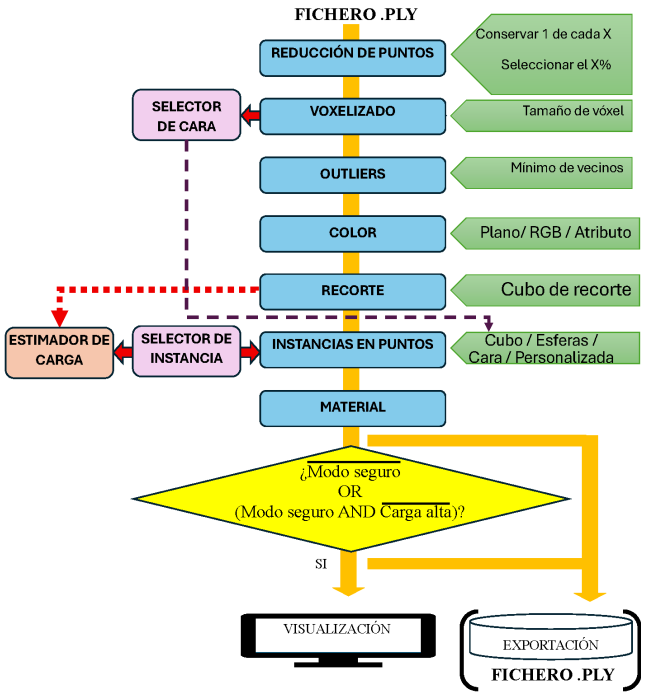


Figura 1 Flujo de funcionamiento del modificador *VisualizadorPuntos* desarrollado. A la nube de puntos del fichero importado `.ply`, se le van aplicando diversos cálculos y modificaciones, en función de los parámetros de usuario (bloques verdes). El resultado final es representado en el editor de Blender (si se cumplen las condiciones definidas por el estimador de carga) y puede ser exportada opcionalmente a un nuevo fichero `.ply`.

- **Muestreo periódico por índice:** Realiza un diezmado regular basado en el orden de almacenamiento de los datos primitivos. Evaluando el índice entero i de cada punto de entrada frente a un factor de paso periódico $k \in \mathbb{Z}^+$, el subconjunto de puntos seleccionados S se define formalmente mediante la condición del operador módulo:

$$S = \{ p_i \mid i \bmod k = 0 \} \quad (2)$$

Estas operaciones no persiguen una simplificación geométrica óptima bajo criterios de curvatura, sino proporcionar un mecanismo para ajustar la densidad visual y el rendimiento computacional.

2.2. Voxelización de la nube de puntos

La voxelización transforma la nube de puntos en una estructura volumétrica regularizada. En lugar de representar e instanciar de forma aislada cada coordenada original, esta etapa particiona el espacio 3D en una retícula indexada de celdas cúbicas de lado h, donde h define la resolución o tamaño de vóxel. Desde un punto de vista matemático, la operación de voxelización mapea cada punto de la geometría de entrada con coordenadas cartesianas $p_i = (x_i, y_i, z_i)$ en un vector de índices discretos $v_i \in \mathbb{Z}^3$ mediante la siguiente función:

$$v_i = \left(\left\lfloor \frac{x_i}{h} \right\rfloor, \left\lfloor \frac{y_i}{h} \right\rfloor, \left\lfloor \frac{z_i}{h} \right\rfloor \right) \quad (3)$$

Un valor de h reducido preserva las discontinuidades y la silueta del objeto a costa de incrementar muy significativamente, en un orden $\mathcal{O}(n^3)$, el número de celdas activas y, por tanto, las necesidades de computación y almacenamiento en memoria. Por el contrario, un valor de h elevado simplifica la distribución espacial o geometría aparente, dando lugar a representaciones abstractas y esquemáticas idóneas para su integración en entornos interactivos de bajo rendimiento.

La voxelización actúa como un mecanismo de regularización y discretización homogénea, ya que, si varios puntos 3D caen dentro de los límites geométricos de una misma celda indexada v_i, se reubican en el centro de la celda y se fusionan mediante el nodo Merge by Distance. Esta transformación permite atenuar irregularidades de posicionamiento de escala inferior al vóxel, reduciendo el número de datos necesarios para representar la escena.

2.3. Recorte, tratamiento de color y filtrado de outliers

La herramienta incorpora módulos auxiliares orientados a realizar recortes espaciales, aplicar distintos modos de coloreado y eliminar puntos aislados u *outliers*.

El módulo de recorte espacial permite limitar la visualización de la nube a una región definida por la malla de la escena que se seleccione. El volumen efectivo de recorte se obtiene a partir de la caja envolvente de esa malla. Para ello, se calcula su *bounding box* y se comparan las coordenadas de cada punto de la nube con los valores mínimos y máximos de la caja en las direcciones X, Y y Z. Los puntos situados fuera del intervalo definido por esta caja son eliminados del flujo de visualización cuando el recorte está activado. Este mecanismo permite aislar regiones de interés, generar secciones parciales de la nube o limitar la escena antes de la instanciación de primitivas.

El tratamiento del color se organiza en tres modos de color:

1. **Color plano:** todos los puntos reciben un color uniforme definido por el usuario.
2. **Color RGB original:** se recupera el atributo cromático original de la nube, habitualmente denominado `Col` en archivos `.ply`.
3. **Color por atributo personalizado:** el usuario introduce el nombre de un atributo escalar asociado a los puntos. El sistema calcula automáticamente el color que corresponde a cada valor escalar, usando un mapa de color RYGB (Red-Yellow-Green-Blue). De este modo, magnitudes escalares como distancia, tiempo, intensidad u otros campos asociados a la nube pueden representarse de forma visual dentro de Blender.

El módulo de color incluye además una opción para eliminar valores nulos.

Por último, el filtrado de *outliers* basado en vecindad local es un filtro que no debe interpretarse como un método estadístico avanzado, sino como un procedimiento geométrico sencillo para detectar puntos aislados. Para cada punto, la herramienta evalúa la presencia de vecinos en direcciones discretas alrededor de su posición, empleando una distancia de detección ajustable. El número de vecinos detectados se compara con un umbral mínimo definido por el usuario. Si el número de vecinos es inferior a dicho umbral, el punto se clasifica como *outlier* y puede ser eliminado del flujo geométrico.

2.4. Instanciación de primitivas geométricas y selector de caras

Una vez definida la geometría de entrada, ya sea a través del muestreo espacial directo de la nube o de su retícula voxelizada, el sistema instancia una malla tridimensional sobre cada coordenada. Matemáticamente, dada una primitiva base G_0, la geometría instanciada G_i correspondiente a una posición p_i se obtiene mediante la composición de una transformación de escala $S(s)$, dependiente del parámetro introducido por el usuario, y una traslación $T(p_i)$, tal que:

$$G_i = T(p_i)S(s)G_0 \quad (4)$$

Esta operación permite alternar interactivamente entre diferentes tipologías de primitivas, G_0, desde cubos perfectos, idóneos para visualizar estructuras voxelizadas, hasta esferas de baja resolución (esferas icosaédricas o esferas UV) y caras planas orientadas. Este enfoque modifica radicalmente la apariencia de la escena sin alterar ni destruir la nube de puntos procesada.

Como última operación para la visualización, el selector de caras permite controlar la representación mediante caras orientadas según direcciones principales. Para ello, el sistema evalúa la existencia de vecinos en una dirección determinada y permite seleccionar caras asociadas a direcciones positivas o negativas de los ejes X, Y y Z. Esta funcionalidad resulta útil cuando se desea representar superficies expuestas de una nube voxelizada mediante caras planas, reduciendo la presencia de geometría interna no visible.

2.5. Estimación de carga visual y modo seguro

La elección de la primitiva y el número de puntos procesados influyen directamente en la complejidad final de la escena. Por ello, la herramienta incorpora una estimación de carga visual orientado a advertir o limitar configuraciones potencialmente costosas para el visor de Blender, que pueden comprometer la estabilidad del sistema.

El módulo preventivo de carga previene de un excesivo consumo de recursos computacionales que impida la manipulación interactiva de las nubes de puntos. Lo que se hace, de forma asíncrona, es hacer una estimación analítica de la carga visual esperada C_{est} Esta variable se calcula como el producto entre el número de puntos de la geometría, N_r por el coste poligonal específico asociado a la primitiva activa C_p:

$$C_{est} = N_r \cdot C_p \quad (5)$$

Este valor se compara con un límite de seguridad estático C_{lim} predefinido. Si se verifica que $C_{est} > C_{lim}$, el sistema entra en "Modo Seguro" y se interrumpe el flujo hacia el motor de renderizado. Simultáneamente, el script de automatización acoplado al nodo emite una notificación de advertencia en el panel de propiedades del usuario, previniendo saturaciones críticas de la memoria gráfica.

3. Resultados

El uso de la herramienta parte de un archivo en formato `.blend` que contiene el modificador *Geometry Nodes* desarrollado, junto con una nube de puntos de prueba. En primer lugar, el usuario puede activar dicha nube o bien importar un archivo `.ply` propio. Una vez cargada la nube en la escena, tendrá que añadirse un modificador *Geometry Nodes* a la nube. A continuación, en la pestaña *Geometry Nodes*, se deberá asociar el árbol de nodos *VisualizadorPuntos*. A partir de ese momento, los diferentes parámetros, que se muestran como bloques verdes en la Figura 1, quedan disponibles desde el panel del modificador.

La herramienta mantiene un carácter no destructivo, es decir, la nube de puntos original, $\mathcal{P}$, se conserva como geometría de partida, mientras que las distintas representaciones se generan mediante la evaluación procedimental del modificador. Esto permite comparar configuraciones, ajustar parámetros de procesamiento y preparar diferentes salidas visuales sin modificar el conjunto de datos original.

Las pruebas funcionales se realizaron en Blender 5.1 sobre un equipo portátil con Windows 11, procesador Intel Core i7-10750H, 32 GB de RAM y tarjeta gráfica NVIDIA RTX 2060. Se emplearon escaneos reales procedentes de escáner láser terrestre, nubes sintéticas generadas mediante *TLSynth* (Pérez *et al.*, 2025) y conjuntos de datos disponibles en repositorios abiertos (GlobalDigitalHeritage, 2019) (Flynn, 2017). El objetivo no fue comparar el rendimiento con bibliotecas especializadas, sino comprobar el funcionamiento integrado de los módulos de reducción, voxelización, coloreado, recorte, instanciación y control de carga visual.

En primer lugar, se evaluó la reducción de densidad y voxelización. La Figura 2 muestra el efecto de conservar distintos porcentajes de una nube, lo que permite ajustar el compromiso entre nivel de detalle y fluidez de visualización. Esta operación resulta especialmente útil como primera etapa cuando el número de puntos dificulta la navegación interactiva en el visor. No obstante, debe tenerse en cuenta que la reducción actúa sobre los puntos de la nube, mientras que la geometría visualizada puede aumentar si se activa la instanciación de primitivas: por ejemplo, cada punto representado mediante un cubo genera una malla de 8 vértices, y cada cara plana una malla de 4 vértices. Por tanto, el coste visual final depende tanto del número de puntos conservados como de la primitiva seleccionada.

La Figura 3 recoge un caso de uso con una nube que originalmente tiene 210 millones de puntos procedente de escaneos láser. A la izquierda se ve el caso de una reducción con una selección porcentual del 15%, que da lugar a una geometría de 60.000.000 puntos. En el centro y a la derecha se combina una selección del 25 % y del 90 % junto con una voxelización de resolución 0,05 m, que generan, respectivamente, una nube de 25.000.000 y 29.000.000 puntos. En todos los casos, se han instanciado cubos para la visualización en color de la nube.

Aunque ambas configuraciones reducen la complejidad respecto a la nube original, la voxelización permite obtener una representación espacialmente más regular y manejable

En lo que respecta a la voxelización y el color, se probó con una nube de puntos producida a partir de la extensión TLSynth. La Figura 4 muestra la aplicación del modificador sobre una nube de 53.699.232 puntos. La imagen inicial, sin modificador, se compara con tres representaciones voxelizadas con

diferentes tamaños de vóxel y modos de color (RGB, atributo 'Distancia' y atributo 'Tiempo').

El recorte espacial y el filtrado de *outliers* se muestran en la Figura 5. En este ejemplo, el recorte se realiza a partir de la caja envolvente del objeto seleccionado (se utiliza el cubo de recorte incluido en el archivo `.blend`) eliminando del flujo los puntos situados fuera de dicho volumen. Complementariamente, el filtrado de *outliers* permite eliminar puntos aislados mediante un criterio local de vecindad.

Figura 2: Aplicación de reducción a la nube basada en la selección del 50 %, 15 % y 1 % (de izquierda a derecha).

Por otro lado, se probó en las diferentes nubes la instanciación de diferentes primitivas y su influencia en el rendimiento de visualización. La Figura 6 ilustra la instanciación de diferentes primitivas geométricas sobre la nube procesada. La misma distribución espacial puede representarse mediante cubos, caras, esferas o una primitiva personalizada, modificando la apariencia final sin alterar la geometría de partida. Esta flexibilidad permite adaptar la visualización al objetivo de la escena: inspección preliminar, representación esquemática, renderizado o integración con otros objetos 3D.

Figura 3 Visualización de una nube obtenida mediante diversos estacionamientos de láser escáner, a la que se le ha aplicado una reducción con selección del 15 % de la nube (izquierda) y una selección del 25 % (centro) y del 90 % (derecha) de la nube con una voxelización de tamaño 0,05 m.

Durante los ensayos se forzó deliberadamente una saturación gráfica reduciendo el tamaño de vóxel y seleccionando primitivas de alta densidad poligonal. El estimador asíncrono detectó la superación del umbral de seguridad, $C_{est} > C_{lim}$, activando el Modo Seguro de la herramienta e interrumpiendo la visualización en Blender, demostrando la eficacia del sistema para prevenir bloqueos de la GPU y mostrando de forma reactiva la alerta en la interfaz del usuario.

Finalmente, la Figura 7 muestra una imagen renderizada obtenida a partir de una nube voxelizada con cubos instanciados. La escena incorpora iluminación de varios colores y configuración de cámara propias de Blender, lo que evidencia una de las utilidades principales de la herramienta: no solo reducir o preparar la nube, sino integrarla en un entorno gráfico completo para generar imágenes, animaciones o composiciones con otros modelos 3D.

4. Conclusiones

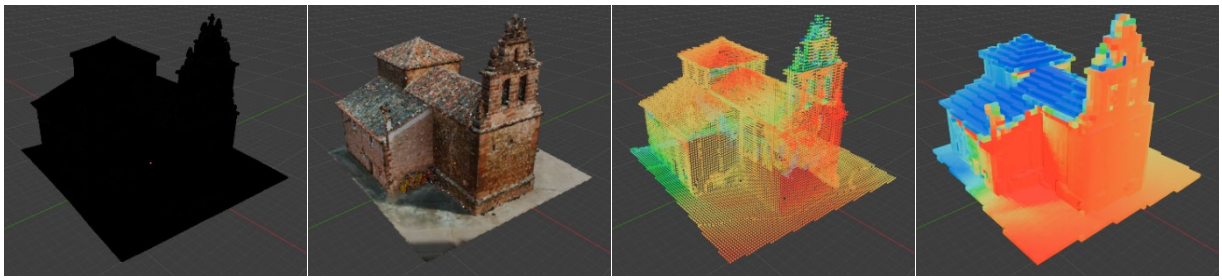
En este trabajo se ha presentado el diseño, implementación y validación de una herramienta procedimental de código abierto para la exploración y generación de representaciones visuales sobre nubes de puntos 3D.

Las pruebas realizadas han permitido comprobar el funcionamiento integrado de los principales módulos de la herramienta: reducción por porcentaje o muestreo periódico, voxelización basada en la cuantización espacial de puntos, recorte mediante la caja envolvente de un objeto de control, visualización de atributos escalares y generación de representaciones mediante primitivas geométricas.

Desde el punto de vista del rendimiento, la principal limitación del sistema radica en trabajar dentro de un entorno gráfico generalista. A diferencia de un software dedicado, que estructura los datos espaciales mediante árboles de partición optimizados, como *octrees* o *KD-trees*, la evaluación procedimental en *Geometry Nodes* puede presentar cuellos de botella con nubes de alta densidad. No obstante, la integración del estimador de carga visual (C_{est}) y del modo seguro permite reducir el riesgo de ralentización o bloqueo del visor cuando se seleccionan configuraciones excesivamente costosas, especialmente al combinar grandes cantidades de puntos con primitivas de elevada complejidad geométrica.

El desarrollo y evaluación de esta herramienta pone de manifiesto el potencial de Blender para integrar operaciones básicas de procesamiento y representación gráfica de nubes de puntos mediante programación visual. A diferencia de las bibliotecas de procesamiento numérico especializadas (como PCL u Open3D), orientadas al análisis algorítmico y al procesamiento avanzado, el modificador propuesto se sitúa en un ámbito complementario: permite reducir, voxelizar, recortar, colorear e instanciar nubes de puntos dentro de un entorno 3D interactivo, aprovechando, además, sus capacidades para iluminación, cámara, renderizado y composición de escenas.

Como líneas de trabajo futuro, se plantea mejorar la integración con formatos habituales de nubes de puntos, como LAZ o E57, para reducir la dependencia de conversiones intermedias. Adicionalmente, se estudiará la posibilidad de exportar atributos y máscaras de segmentación semántica. También se planea realizar una evaluación cuantitativa sistemática del rendimiento de la herramienta, considerando métricas como el tiempo de actualización del modificador, el consumo de memoria gráfica y los límites prácticos de uso en función del número de puntos, la resolución de vóxel y la complejidad de la primitiva instanciada. Por último, se prevé convertir la herramienta en una extensión de Blender fácilmente distribuible e instalable por otros usuarios.



(a) Sin modificador – 53699232 puntos.

(b) Voxel 0,1 m – Color RGB – 1311152 puntos (cubos 0,1 m)

(c) vóxel 0,25 m – Color Atributo 'Distancia' – 219520 puntos (cubos 0,1 m)

d) vóxel 0,4 m – Color Atributo 'Tiempo' – 85432 puntos (cubos 0,4 m)

Figura 4: Visualización de la nube de puntos en Blender antes de la aplicación del modificador (a) y después de la aplicación (b,c,d), con diferentes tamaños de vóxel, color y tamaño de cubo instanciado.

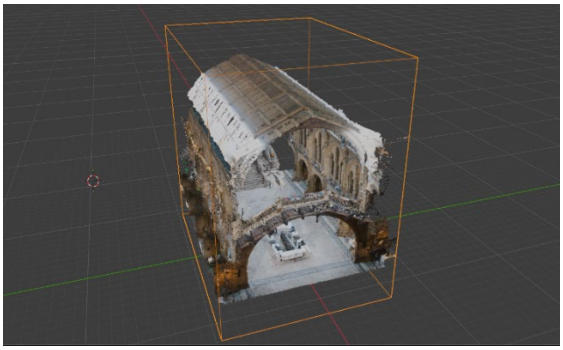


Figura 5: Ejemplo de recorte aplicado a una nube, a la que se le aplicado también el filtrado de outliers, utilizando el cubo de recorte incluido en el archivo .blend.

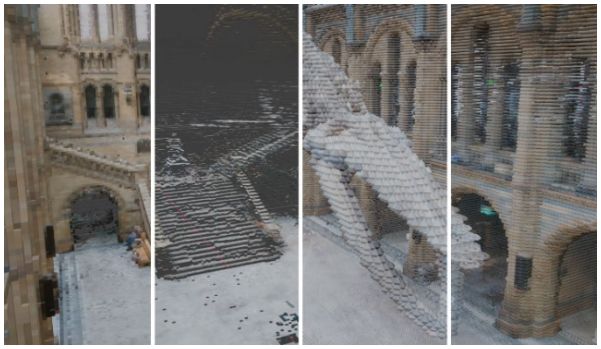


Figura 6: Representación de la nube con diferentes primitivas instanciadas: cubo, cara, esfera UV y personalizada – Torus (de izquierda a derecha).

Figura 7: Imagen renderizada de una nube de puntos voxelizada, con cubos instanciados en cada punto, y en cuya escena se han añadido diversos puntos de luz de varios colores e intensidades. Adicionalmente se ha integrado en la escena una malla 3D de una lámpara de techo.

Agradecimientos

Este trabajo ha sido financiado por la Consejería de Educación, Ciencia y Formación Profesional de la Junta de Extremadura a través de la Ayuda a Grupos de Investigación GR24059, cofinanciada por Fondos FEDER.

Referencias

Blender Foundation, 2025. Blender (version 4.x). Free and Open-Source 3D Creation Suite. Available at: https://www.blender.org (accessed: 04/06/2026).

Bullinger, S., Bodensteiner, C., Arens, M., 2021. A photogrammetry-based framework to facilitate image-based modeling and automatic camera tracking. Proceedings of the International Conference on Computer Graphics Theory and Applications (GRAPP 2021), 163-170.Cignoni, P., Callieri, M., Corsini, M., Dellepiane, M., Ganovelli, F., Ranzuglia, G., 2008. MeshLab: an Open-Source 3D Mesh Processing System. ERCIM News, 73, 45–46. DOI: 10.5220/0010319801060112

Flynn, T., 2017. Hintze Hall, NHM London [point cloud, 3D model]. Sketchfab. Available at: https://skfb.ly/6sXWG (accessed: 04/06/2026)

Girardeau-Montaut, D., 2016. CloudCompare (version 2.x): 3D point cloud and mesh processing software. EDF R&D, Telecom ParisTech, France. Available at: http://www.cloudcompare.org (accessed: 04/06/2026).

GlobalDigitalHeritage, 2019. La Barbolla Church, Siguenza Spain [3D model]. Sketchfab. Available at: https://skfb.ly/oNWyX (accessed: 04/06/2026).

Gumster, J., Lampel, J., 2022. Procedural Modeling with Blender's Geometry Nodes: A workshop on taking advantage of the Geometry Nodes feature in Blender for procedural modeling. In ACM SIGGRAPH 2022 Labs (SIGGRAPH '22). Association for Computing Machinery, New York, NY, USA, Article 5, 1–2.
DOI: 10.1145/3532725.3538516

Lucke, J., 2015. *Animation Nodes* (Versión 2.x). Available at: https://github.com/JacquesLucke/animation_nodes (accessed: 04/06/2026)

Nortikin, 2013. *Sverchok: Parametric tool for Blender* (Versión 1.x). Available at: https://github.com/nortikin/sverchok

Pérez, E., Sánchez-Hermosell, A., Merchán, P., 2025. TLSynth: A Novel Blender Add-On for Real-Time Point Cloud Generation from 3D Models. Remote Sensing 17(3), 421.
DOI: 10.3390/rs17030421

Pérez, E.; Merchán, P.; Espacio, A.; Salamanca, S., 2024. Fusion of Thermal Point Cloud Series of Buildings for Inspection in Virtual Reality. Buildings. 14(7), 2127.
DOI: 10.3390/buildings14072127

Rusu R. B., Cousins, S., 2011. 3D is here: Point Cloud Library (PCL). IEEE International Conference on Robotics and Automation, Shanghai, China, 2011, pp. 1–4.
DOI: 10.1109/ICRA.2011.5980567.

Uhlík, J., 2017. Point Cloud Visualizer for Blender (Versión 3.x). Available at: https://jakubuhlik.com/docs/pcv/ (accessed: 04/06/2026).

Zhou, Q.-Y., Park, J., Koltun, V., 2018. Open3D: A Modern Library for 3D Data Processing. arXiv preprint arXiv:1801.09847.